(54) Title: MACHINE VISION SPREADSHEET

(57) Abstract: A user interface is provided for programming machine vision applications that includes an image window for displaying an image to be analyzed, and a spreadsheet for analyzing the image, displayed with the image window. A hand-held control pad can be used to navigate over the spreadsheet,

[Continued on next page]

thereby selecting cells of the spreadsheet. The control pad can also be used to select items from menus, and select alphanumeric characters as parameters for input into the spread sheet. The menus include various software tools and functions, such as vision functions. The spreadsheet is semi-transparent, such that the image can be seen through the spreadsheet. The spreadsheet is adapted to perform conditional cell execution. The spreadsheet can also include functions that can operate on values stored over a time interval. When the control pad indicates a particular cell of the spreadsheet, a change occurs in the appearance of the image displayed. The spreadsheet can also include cells adapted to buffer outputs, thereby providing circular reference for recursive operation, and can include cells that can be assigned arbitrary depth in a time dimension. Also, complex vision objects can be instantiated as a single cell in the spreadsheet, the single cell having data that is accessible through a plurality of member functions. Moreover, the use interface includes a spreadsheet mask for obscuring portions of the spreadsheet, leaving portions unobscured that can be used to control operation of the spreadsheet. The user-interface is particularly advantageous for machine vision applications, and other applications using large data sets.

Machine Vision Spreadsheet

Cross-references to Related Applications

This application is a continuation-in-part application of application No. 09/370,705, filed 8/9/99, application No. 09/370,706, filed 8/9/99, and application No. 09/370,808, filed 8/9/99, and claims the benefit of priority of provisional patent application No. 60/160,958, filed 10/22/99, and provisional patent application No. 60/169,514, filed 12/7/99.

Field of the Invention

This invention relates generally to software user-interfaces and programming environments, particularly as applied to industrial automation.

Background of the Invention

In the field of factory automation, the high cost of implementing machine vision technology represents a significant barrier to the wide-spread adoption of industrial vision. Traditionally, a large part of this cost has been attributed to the expense of software development. Programmers with specialized machine vision experience are often required to accomplish ostensibly simple machine vision tasks. Although many attempts have been made to simplify application development, there is no clear solution to the machine vision ease-of-use problem.

In general, the ease-of-use problem is concerned with the natural tradeoffs between flexibility and simplicity. As any computer interface becomes more flexible, it becomes more complex and as such more difficult to use. Conversely, simplifying the interface generally limits its range of applicability.

With the advent of windows-oriented operating systems and graphic user interfaces, most machine vision companies have made vision algorithms

1

accessible from interactive graphical interfaces implemented in industry-standard shared libraries such as DLL, VBX, COM, and ActiveX. Other machine vision suppliers have developed dedicated menu-driven interfaces. Still others have created elaborate graphical programming environments based both physical and abstract connection metaphors. None of these paradigms have proven completely satisfactory.

Library interfaces, even those offering active graphical components, are essentially programming accessories. The user must still construct an intricate sequence of instructions linking the various vision primitives into a single application. This process requires a skilled software developer.

Menu-driven systems do not require conventional programming but are too inflexible to be considered generally purpose. These systems are limited to narrow application domains where all aspects of an application setup and configuration can be anticipated.

Graphical programming seems like an improvement over conventional programming, but in reality the arcane nature of the metaphor represents a substantial challenge to an unskilled user. Connecting wires to boxes may be a more playful way to generate code, but the true ease-of-use advantages are negligible.

The superior ease-of-use characteristics of conventional electronic spreadsheets are well-known, especially in the field of office automation. The use of electronic spreadsheets in other fields is also known, although in the past, certain computer applications have been unable to take full advantage of the spreadsheet metaphor. One reason for this is that a large class of computer applications require algorithms that operate over time-based intervals of a continuous data stream, and conventional spreadsheets provide no intrinsic support for continuous data processing over more than one processing interval.

An electronic spreadsheet is, essentially, a means of graphically representing a set of expressions as a grid of cells. Each cell in the spreadsheet grid represents a parenthetical expression that can, in turn, be a function of some number of other cellular expressions.

2

A spreadsheet program updates its grid, as necessary, to maintain the programmed relationship between cellular expressions. Electronic spreadsheets are intended to provide immediate response to any modifications of the programmed expressions. However, in conventional electronic spreadsheets, cellular expressions have no means for accessing previously evaluated results from prior processing intervals. Absent this capability, it is impossible for a conventional spreadsheet to process a continuous data stream of incoming data on a time-based interval of a duration greater than one processing interval.

Although methods for implementing algorithms that operate on continuous data streams are well-known in computer programming, no methods are known for implementing this class of processing within the context of an electronic spreadsheet.

Machine vision programming tasks typically involve the manipulation of images and other large data sets, a task poorly suited to conventional electronic spreadsheets.

A standard spreadsheet consists of a rectangular grid of cells, each cell representing a single value that is, in turn, a function of some number of other cells in the grid. The standard electronic spreadsheet is suited for implementing formulas that require a relatively small number of input values and generate a single scalar output. In contrast, certain machine vision functions are parameterized by arguments representing a million or more pixel values. Since there is no facility in a standard electronic spreadsheet that enables a single cell to represent multiple values, the standard spreadsheet model is clearly not suited to support functions that operate on images and other large data sets.

In the fields of scientific visualization and computer graphics, the term "spreadsheet" has been loosely applied to certain programs that manage multiple images and other data representations in a two-dimensional rectangular grid within a bounding window. These programs, typically designed for graphical editing and array processing, essentially, allow individual data cells to be formatted as data arrays. However, in order to accommodate data sets of varying size, these interfaces generally abandon the strict two-dimensional grid relationship between cells that is the essence

3

of the spreadsheet paradigm. In fact, most of these programs are unrecognizable as spreadsheets, and are generally unsuited for machine vision.

In the past, many computer applications have been unable to take advantage of the spreadsheet metaphor. One reason for this may be that computer applications are often required to exhibit conditional statement execution behavior. However, conditional statement execution behavior is beyond the capability of conventional electronic spreadsheet programs.

In the field of pattern recognition, for example, it is common for a single program to execute one set of instructions during training, and another set of instructions during classification, the instructions for classification being dependent on exemplar data extracted during training. In this field it is also common for programs to dynamically adapt their order of execution based on external control parameters. Consequently, known electronic spreadsheets cannot adequately be used for programming applications in the field of pattern recognition.

In conventional programming, positional order of program statements and conditional branching within program statements drive the order of execution of the program statements. In the BASIC programming language, for example, a program statement at line 20 in a program listing is always executed after the program statement at line 10, unless a branch statement is encountered and a jump instruction is invoked.

Unlike a programming language, an electronic spreadsheet is essentially a means for graphically representing a single expression by distributing the expression across a rectangular grid of cells (or any regular array of cells), each cell being a parenthetical expression that can be a function of some number of other expressions in cells across the grid. As in typical expression evaluation, precedence and dependencies drive the order of execution. Since the order of execution is not explicitly programmed, there is no spreadsheet equivalent to a jump instruction in conventional spreadsheets. This limitation of conventional spreadsheets makes programming certain kinds of behavior difficult, if not impossible.

Conventional electronic spreadsheets are known to support limited conditional behavior, e.g., conditional argument value assignment using the

4

standard IF functions.. An IF function consists of three arguments: A, B, and C, where argument A is a logical switch expression, and arguments B and C are the two possible expressions to be evaluated. If the expression in argument A evaluates TRUE, the expression in argument B is evaluated and returned. If the expression in argument A evaluates FALSE, the expression in argument C is evaluated and returned.

Like IF, the CHOOSE function simply evaluates and returns one of N expressions. The evaluation of the expression in the first argument of the CHOOSE function determines which of the N expressions will be evaluated and returned, which is essentially a generalization of the IF function. However, the functions IF, CHOOSE, and similar functions, are only capable of selecting among result values, and are incapable of controlling the order of execution of expressions associated with a cell or a set of cells within the spreadsheet.

Summary of the Invention

Overcoming the problems of the known ease-of-use approaches, the invention introduces a spreadsheet environment and programming model, powerful enough to be applied to a wide range of problems, yet accessible to one who typically does not have programming experience or expertise, such as a typical manufacturing engineer.

Although spreadsheets are widely used in manufacturing, they are not available as a framework for use in industrial computer vision.

Spreadsheet ease-of-use derives from the almost purely modeless character of its easily mastered intelligent worksheet metaphor. Unlike conventional programming, the development environment, run-time environment, and user interface are all one and the same. In the spreadsheet, there are no abstract modes; what you see is truly what you get. Change any number or any formula in any cell and everything else updates automatically.

The challenge met by the invention was to develop a spreadsheet-programming environment for industrial machine vision. Ideally, this would

have been accomplished by simply finding an existing spreadsheet engine and adding vision functions. Unfortunately, standard spreadsheets are inadequate for industrial machine vision. Achieving the desired result required substantial extensions to the fundamental spreadsheet model.

One aspect of the invention is a spreadsheet interface for programming industrial machine vision applications. The machine vision spreadsheet of the invention offers many key improvements over conventional spreadsheets, as detailed herein below.

Semitransparent Worksheet Superposed on Image

The user interface of the invention seamlessly integrates the worksheet and the image. The worksheet grid appears to be a semitransparent sheet situated on top of an image display. This technique lets the user simultaneously view a full screen image and the associated worksheet. Alternative techniques include displaying the image and the worksheet side-by-side or supplying a means to flip from one display to the other. Both of these alternatives have significant drawbacks.

Locating the image beside the worksheet creates display size limitations. On small monitors, common to most production environments, both the image and the worksheet would be small and difficult to see. Such an interface would probably need to scroll the image window or scale it into a low-resolution format. Scrolling techniques that hide large sections of the image tend to cause confusion, since it is never quite clear which section of the camera's field of view is being displayed. Scaling the image into a low-resolution window is also undesirable, since graphics overlays such as crosses and region boxes cannot be positioned with single pixel precision.

Making the image and worksheet displays mutually exclusive presents an important ease-of-use limitation. This technique makes it difficult to reinforce the relationship between the contents of the image and the contents of the worksheet. In the ViewPoint user interface, changing the contents of the image can lead to changes in the contents of the worksheet. Similarly, changing the contents of the worksheet can lead to changes in the image.

6

When the image and the worksheet display are mutually exclusive, the relationship between cause and effect becomes detached. In other words, it becomes difficult for the user to associate actions taken in the worksheet with results appearing in the image.

In preferred technique—the one employed by the user interface aspect of the invention—a semitransparent worksheet is superimposed on a full resolution image display. This approach supplies immediate feedback for user-initiated actions executed in either domain. When necessary, the image can still be displayed exclusively, but this can be reserved for times when the image demands system modal focus.

Remote Input Device Suitable for a Manufacturing Environment

Most spreadsheet user-interfaces run on business workstations, typically personal computers. The manufacturing environment for machine vision is far different and far less accommodating than the typical office environment. Users are often standing, with no desk for a keyboard and mouse. In addition, operators may need to be several feet away from the computer monitor to adjust the trigger, the camera, the lights, and other remote components.

The user interface aspect of the invention better accommodates the manufacturing environment. Its pendent remote interface device lets operators control the vision system while standing at a distance from the monitor, even while wearing gloves. Operators navigate using a combination of four direction buttons and four action buttons. Unlike any other spreadsheet interface or environment, an entire application can be constructed using a total eight buttons. Although it somewhat less familiar than the keyboard/mouse combination, it is, nonetheless, well suited for its intended purpose and far superior to conventional input devices, given the practical constraints imposed by the manufacturing environment.

Worksheet Focus Coordinated with Image Display

As explained above, the user-interface aspect of the invention strives for the seamless integration of the spreadsheet and the image. It achieves this partly by simultaneously displaying image and worksheet data. Beyond simple

7

overlaying data, the invention coordinates browsing in the spreadsheet with presentation of the image. The spreadsheet cell that has focus—that is, the cell selected—can automatically cause a change in the image display consistent with the associated worksheet formula. The feature is also referred to as "flyover", in that as the point of focus "flys over" the cells of the spreadsheet, the associated image changes in some way. The image is typically visible through the spreadsheet displayed as a semi-transparent (translucent) image. For example:

When a cell with an image-related input parameter is selected, the associated parameters are graphically depicted in the associated image.

When an image-processing cell is selected, the invention automatically displays the processed image.

When a graphics cell is selected, the associated graphic component will be activated in the associated image. If it is already visible, its color will be changed so that it can be uniquely identified.

2D Image Integrated in Dependency Tree

In most programming languages, positional order and conditional branching drive the order of execution. In BASIC, for example, line 20 is always executed after line 10, unless a branch is encountered and a GOTO LINE# statement is invoked. Unlike a conventional programming language, a spreadsheet is essentially an elaborate representation of a single expression. As in expression evaluation, precedence and dependencies drive the order of execution.

For example, in the expression X=A*(B+C), the parenthetical expression (B+C) is always executed first. This is true even if the order with respect to the factor A is reversed, as in X=(B+C)*A. The parentheses enclosing B+C indicate to the expression evaluation engine that adding takes precedence over the multiplication. Taking the example a step further, in the expression X=A*((B+C)*(D+E)), (B+C) and (D+E) are executed first, but the order of execution is unspecified and unimportant.

8

In a spreadsheet, any expression contained within a cell is the functional equivalent of a parenthetical expression. So, if a cell Z contains B+C, cell Y contains D+E, and cell X contains A*Y*Z, then cell X is functionally equivalent to the expression X=A*((B+C)*(D+E)), since cells Y and Z must be satisfied before cell X can be evaluated. Spreadsheet execution is driven by dependency, not by the order or position of the active cells. The spreadsheet aspect of the invention makes important contributions to the known spreadsheet paradigm by introducing, among other aspects, two-dimensional image data, complex data structures, and objects into the dependency graph.

Most known spreadsheet interfaces allow users to insert bitmap image data in the worksheet, but this is mostly for simple display. The vision spreadsheet interface of the invention is unique in its integration of two-dimensional images with worksheet formulas. Unlike most spreadsheets, which deal with images by simply overlaying bitmaps on top of cells, the invention represents an entire image with a single cell. According to an aspect of the invention, an image acquisition initiates a chain of execution starting with those cells that reference the acquisition buffer as an input data. If an image feature extraction operation depends on some form of image preprocessing, then a reference to the preprocessed image cell can be passed as an argument to the image feature extraction function.

More concretely, suppose the objective is to inspect the contrast of a preprocessed image region. The end result is pass/fail flag, where 1=PASS and 0=FAIL. One could write the pseudo-expression:

OUTPUT(Bit#,CONTRAST(HISTOGRAM(IMAGEPROCESS(ACQBUF
FER)))>Thresh)

As in the previous example, this expression could be split into individual cells.

CELL A= IMAGEPROCESS(ACQBUFFER)
CELL B= HISTOGRAM(A)
CELL C= CONTRAST(B)

9

CELL D= Thresh

CELL E= OUTPUT(Bit#,C>D)

When the system of the invention receives an acquisition trigger, the new image data drives the update of (Cell A). (Cell A) creates a processed image represented by (Cell A). These events force the update of (Cell B), which represents the grayscale histogram of (Cell A). After the histogram is computed, the image memory associated with (Cell A) can be automatically released, since all dependencies associated with the processed image have been satisfied. (Cell C) reduces the contrast to a single value. (Cell E) compares that value to a threshold stored in (Cell D) sending the Boolean result to a specific bit of discrete I/O.

In addition to simple cell dependencies, the spreadsheet of the invention understands and manages regions of interest within image buffers. Data written to a specific section of an image buffer represented by a single cell will update only those cells dependent on the pixels written. If in this example, an acquisition were directed to a region of interest that doesn't intersect with the image-processing region, the chain of execution would not be initiated. This is because there would have been no actual change of pixel input data with respect to the image-processing operation in (Cell A).

Expressions can Depend on Clock Input

As described above, spreadsheet execution is driven by the need to update dependent expressions. According to the spreadsheet of the invention, however, expressions may be dependent on external clock input. Therefore, broadcasting a clock input to the spreadsheet causes clock dependent cells to update. A clock signal of the invention can be the result of a manual trigger action, a free running clock, a timed internal clock, or an external trigger. The presence of this clock dependency gives the spreadsheet of the invention unique advantages, allowing the extension of spreadsheet techniques into the time domain.

Expressions Respond to Coded Triggers

As described above, all cells and expressions respond to global external clock input. Cells respond to this clock signal in one of three ways: Run, Define, or Refine.

The Run response is the normal spreadsheet response: cells simply execute their nominal function.

The Refine signal tells cells that react to Refine to take the image and worksheet data existing from the previous clock step and use it to update their internal state. For example, a search might take the most recent POSE result coordinates and use them isolate image data for the purposes of updating a stored model template.

The Define signal tells cells that react to a Define to execute a Define process. For example, the search function might respond by automatically bringing up the search property sheet and prompting the user to train a new model template.

This unique ability to respond to coded triggers allows the spreadsheet of the invention to react much as if it were a single vision tool. The user of the spreadsheet of the invention, using this capability, can develop not only the run-time sequence of operation but also specify the pattern of both manual and automatic training behavior.

Special Cells Buffer Outputs for Recursive Operation

As described above, all cell expressions implicitly depend on clock input. This statement is not intended to imply that a clock signal is required to achieve output. The clock is simply one of, potentially, several inputs that must be satisfied.

A conventional spreadsheet does not employ the concept of a globally synchronized system clock. Outputs occur as data propagate freely along a chain of dependencies. Unlike conventional spreadsheets, the invention supports the concept of clocked cells. A clocked cell acts in two phases of the global clock: In the first phase, inputs are satisfied and an output is

11

generated. In the second phase, the output buffer is copied into the internal buffer.

This architecture makes it possible to construct items such as counters and accumulators—something impossible in a known spreadsheet.

For example, suppose one wanted to count defects. Let the defect be asserted by a (1) TRUE output in (Cell A). Let the counter be in (Cell B). In the spreadsheet of the invention, (Cell B) could contain the expression B=B+A— that is, take the contents of (Cell A), add it to the contents of (Cell B), and leave the result in (Cell B). This syntax is legal only because (Cell B) is a clocked cell. On the first phase of the clock signal, (Cell B) copies its contents into a local buffer. On the second phase, (Cell B) accesses its own buffer and (Cell A), sums the two values, and places the result in (Cell B)'s output. Since clocked cells update only on the external clock signal, the circular reference is interrupted by the clock input signal parameter, as illustrated below.

| | | Defect | Defect Counter | |
| | | Cell A | Cell B | |
Clock		A	Bo	Bb
		0	0	0
0	phase 1	1	1=A+Bb	0
	phase 2	1	1	1=Bo
1	phase 1	0	1=A+Bb	1
	phase 2	0	1	1=Bo
2	phase 1	1	2=A+Bb	1
	phase 2	1	2	2=Bo
3	phase 1	1	3=A+Bb	2
	phase 2	1	3	3=Bo

Table 1

In this example, three (3) defects occur in four (4) clock cycles. A conventional spreadsheet cannot duplicate this simple behavior. Clocked cells are essential for time domain integration.

Special Functions Support Data Buffering over Time

The above discussion describes a simple clocked cell. The spreadsheet of the invention supports another variety of clocked cell by way of the FIFO function, which allows spreadsheet users to create pipeline buffers of arbitrary depth.

12

The FIFO function clocks data through a first-in-first-out shift register. The output is the data from N clock periods ago, where N is the depth of the pipeline. The pipeline can also be tapped at any depth necessary. This represents a unique and powerful function with respect to spreadsheet machine vision. Consider three examples:

To compute a Pass-Fail output based on the average of the last N feature responses, the FIFO function could be tapped to access the previous N responses, which could be averaged and compared to the current value.

To delay the output of a Pass-Fail signal—a common problem on assembly lines, since rejects can rarely be handled at the vision inspection station--the Pass-Fail can be fed into a FIFO of the appropriate depth. The pipeline can also be tapped to support actions at any stage between the inspection station and the final reject station.

To compute the velocity of a feature coordinate, three FIFO pipelines could be created. As the feature is located, the first FIFO inputs X coordinate, the second inputs the Y coordinate, and the third inputs a real-time clock. By appropriately tapping the three FIFOs, it is possible to compute the time-based derivatives of the image feature.

These three tasks are impossible using conventional spreadsheet operations.

A Coherent Hierarchy of Data Reduction

The spreadsheet function interface of the invention has many examples of cells that represent more than one value. For example, template search not only returns the number of templates found, it also buffers the POSE coordinates, the score for each result, and internal state data including the trained template. This kind of cell, which we call a complex cell, is actually more like an object than a function. The spreadsheet of the invention preserves the standard single-value return paradigm by providing data-access member functions. For example, the POSE row coordinate of a search results in A1 is available through the access function PoseRow(A1,0).

An elegant feature of the spreadsheet vision architecture of the invention is that image buffers, like vision tools, are treated as complex objects. Image-buffer access functions are vision tools. The image acquisition buffer is simply a complex cell accessed by vision tools that are in turn accessed by increasingly higher-level access functions. Image data is, therefore, systematically reduced to scalar values that can be manipulated by conventional means.

Sine the spreadsheet paradigm forms the basis of many of the worlds most success computer applications, e.g., Lotus 123 and Microsoft Excel, literally millions of users are already familiar with the concept of worksheet based computer programming. Consequently, another important advantage of the machine vision spreadsheet of the invention is that users in an industrial environment will probably already be familiar with the basic aspects of the user interface of the invention. Remarkably, despite the acceptance of conventional spreadsheets in manufacturing, industrial machine vision products have never successfully adopted spreadsheet-programming techniques. This is due, in large part, to the inability of conventional spreadsheet products to manage the dimension and complexity of the machine vision application domain.

Vision applications introduce many complications including an inherent time dimension, complex data types, and special human interface constraints. The invention addresses these complications with innovative extensions including:

Clocked cells and FIFO functions to extend cell dimensionality in time;

A coherent methodology for manipulating and reducing complex data; and

Special human-interface input means and graphical display techniques.

By adapting the well-known spreadsheet-programming paradigm, the enhanced spreadsheet of the invention succeeds in bringing new flexibility and intuitive ease-of-use to a significant number of industrial automation problems, such as machine vision problems, that could otherwise only be solved through complex conventional programming techniques.

14

The invention also provides an electronic spreadsheet adapted for programming the processing of continuous data streams. The invention also provides means and methods that, to a high degree, preserve the fundamental characteristics of conventional electronic spreadsheets. The invention also provides means and methods, which are uncomplicated and intuitive to use.

These benefits and features are accomplished by extending the spreadsheet paradigm to include apparatuses and methods of clocked data buffering, such as by using shift registers, delay lines, FIFOs, pipelines, and even random access memory. The use of shift registers is preferred because shift registers do not require addressing. Delay lines objects are instantiated (memory is allocated for each delay line object) and assigned to spreadsheet cells (each allocated memory is associated with the coordinates of a cell). A delay line object includes a data buffer configured as a variable length shift register, and a method for clocking data through the buffer. In a preferred embodiment, input data streams supply the clocking signals, and "tap functions" support random access of elements, and other analysis over the buffered interval.

The invention extends the conventional electronic spreadsheet paradigm to include the processing of continuous data streams on finite intervals.

The invention further provides methods for adapting electronic spreadsheets to applications in the field of industrial machine vision, while also substantially preserving the fundamental characteristics of conventional electronic spreadsheets. The invention also makes efficient use of a given display area and resolution, using for example, a semitransparent spreadsheet superimposed over an image of the contents of a data buffer. Further, the invention facilitates and simplifies the manipulation of images and other large data sets, while preserving the ease-of-use, and other advantages and benefits, of the spreadsheet paradigm.

The invention exploits a new class of object-oriented spreadsheet functions that incorporate internal data buffering. In addition, the invention provides methods by which the graphical representation of buffered data sets may be displayed simultaneously with a spreadsheet grid, preferably as a

15

graphics layer positioned beneath a variably transparent spreadsheet grid. A split-screen and dual screen approach can also be employed for displaying the buffered data sets alongside the electronic spreadsheet.

In a preferred embodiment, a conventional spreadsheet is extended to include data buffering functions, herein referred to as "single method objects", because from the perspective of the spreadsheet programmer, there is only one public member function, i.e., only one public method. For example, in accordance with the invention, the familiar "search" tool of machine vision is the public member function of the search object, which includes a search function, as well as data associated with the search function.

A single method object can be instantiated and assigned to a spreadsheet cell within a variably transparent grid superimposed on a graphical display buffer, the contents of the graphical display buffer being determined in accordance with the currently selected cell. If an empty cell, or a cell containing a conventional spreadsheet formula is selected, the image last acquired is stored and displayed in the underlying graphical display buffer. If a cell containing a single method object is selected, a graphical representation of the object data members is stored in the underlying buffer and displayed.

The foregoing methods selectively extend data dimensionality to meet the practical requirements of machine vision processing, while still retaining the fundamental characteristics and ease-of-use of a conventional electronic spreadsheet.

The invention also provides methods and apparatuses for programming conditional execution behavior in an electronic spreadsheet, and therefore methods of programming the order of execution of expressions associated with cells in an electronic spreadsheet. While providing such methods and apparatuses, the invention preserves the fundamental characteristics of conventional electronic spreadsheets to a high degree. Moreover, the methods and apparatuses of the invention for programming conditional behavior in electronic spreadsheets are easy and intuitive to use.

The invention includes implicit conditional wrappers associated with each cell of the electronic spreadsheet, whereby the execution of each cellular expression is conditioned upon the state of an individually assigned logical switch expression included in an associated implicit conditional wrapper.

16

In a preferred embodiment, every expression in every cell is enabled or disabled by an implicit conditional wrapper. This implicit conditional wrapper uses a logical switch expression to control the execution of the cell's expression, i.e., to enable or disable the execution of the cell's expression based on the value of the logical switch expression.

In a preferred embodiment, the logical switch expression is entered into the implicit conditional wrapper of a cell using a pop-up dialog box and an associated pop-up window, for example. The dialog box can then be closed by the user, causing the pop-up window and its entered logical switch expression to disappear along with the dialog box. Closing the dialog box makes the logical switch expression appear to no longer be associated with the cell, thereby making the logical switch expression invisible or "implicit", because the existence and nature of the logical switch expression is not visible to the user solely by looking at the contents of the cell. However, the effects of the logical switch expression become manifest upon execution of the electronic spreadsheet.

The logical switch expression can be any legal spreadsheet expression that can be evaluated to logical TRUE or FALSE. If the logical switch expression evaluates to TRUE, the cell's corresponding expression is evaluated, a new result value is returned, and the cell's internal result buffer is updated accordingly. If the logical switch expression evaluates FALSE, the cell's expression is not evaluated, and the cell's result buffer is returned unmodified.

In a preferred embodiment, individual switch expressions are initialized to a constant logical TRUE. In another preferred embodiment, the spreadsheet user interface supports the interactive assignment of a switch expression to any cell or range of cells within the spreadsheet, using dialog boxes and pop-up windows, for example. A user selecting a "cell state" display mode from a menu after selecting a cell will cause a dialog box to appear, along with an associated window that displays the associated logical switch expression.

The default behavior of the improved electronic spreadsheet is unaffected by the extensions and improvements described above if all individual logical switch expressions are initialized to a constant logical TRUE. In this case, a user can ignore the implicit conditionals. Alternatively, the user can mark

arbitrary sets of cells, and then selectively assign logical switch expressions to program desired electronic spreadsheet behavior beyond the capacity of known electronic spreadsheets, such as providing various orders of cell execution.

The invention efficiently and elegantly extends the conventional spreadsheet metaphor to meet many of the fundamental requirements of conditional expression execution in computer programming.

BRIEF DESCRIPTION OF DRAWINGS

The invention will be more fully understood from the following detailed description, in conjunction with the following figures, wherein:

Fig. 1 is an example illustrating methods of electronic spreadsheet programming for processing continuous data streams of the invention;

Fig. 2 is an example output sequence illustrating the response of the example program of Fig. 1 to a specific input sequence;

Fig. 3 is a diagram illustrating the components of a delay line object; and Fig. 3A is a diagram of the parameters of the delay line object of Fig. 3.

Fig. 4 is a block diagram of a computer system for suitable for practicing the invention;

Fig. 5 illustrates a superposition of graphics layers to obtain the user-interface graphics display of the monitor of Fig. 4 ;

Fig. 6 is a depiction of the graphical user interface of the invention with cell A1 selected of a semi-transparent spread sheet that is superimposed on the image corresponding to the single method object instantiated in cell A1;

Fig. 7 is a depiction of the graphical user interface of the invention with cell A2 selected of a semi-transparent spread sheet that is superimposed on the image corresponding to the single method object instantiated in cell A2;

Fig. 8 is a depiction of the graphical user interface of the invention with cell A3 selected of a semi-transparent spread sheet that is superimposed on the image corresponding to the single method object instantiated in cell A3;

18

Fig. 9 is a depiction of the graphical user interface of the invention with cell A4 selected of a semi-transparent spread sheet that is superimposed on the image corresponding to the image last acquired;

Fig. 10 is a sequence of four instances of a portion of an electronic spreadsheet incorporating the improvement of the invention, each instance including a dialog box and a pop-up window making explicit the logical switch expression of the conditional wrapper associated with each cell of the improved spreadsheet of the invention;

Fig. 11 is a listing BASIC computer program that is the functional equivalent of the spreadsheet of Fig. 10;

Fig. 12 is a flow chart illustrating a method of the invention for programming conditional execution in an electronic spreadsheet;

Figs. 13-16 are flowcharts representing the logic of an unimproved spreadsheet;

Fig. 13 is a flowchart of the main loop of a spreadsheet without the extensions of the invention;

Fig. 14 is a flowchart of the DisplaySheet() module of Fig. 13;

Fig. 15 is a flowchart of the EvaluateSheet (Row,Col) of Fig. 13;

Fig. 16 is a flowchart of the EvaluateCell (u,v) of Fig. 15;

Figs. 17 - 29 provide a hierarchically structured flowchart of an example program supporting methods of the current invention, wherein:

Fig. 17 is a flowchart of the main processing loop;

Fig. 18 is a flowchart of a function to display cell contents;

Fig. 19 is a flowchart of a function to input new formulas;

Fig. 20 is a flowchart of a function to update the spreadsheet;

Fig. 21 is a flowchart of a function to respond to a trigger;

Fig. 22 is a flowchart of a function to display the extended data;

Fig. 23 is a flowchart of a function to display a data cell;

Fig. 24 is a flowchart of a function to execute a single formula;

Fig. 25 is a flowchart of a function to execute dependent formulas;

Fig. 26 is a flowchart of a function to execute conventional formulas;

Fig. 27 is a flowchart of a function to execute extended formulas;

Fig. 28 is a flowchart of a function to allocate cell data; and

Fig. 29 is a flowchart of a function to de-allocate cell data.

Detailed Description of Preferred Embodiments

Referring to Fig. 1, a sequence of user interface screens 1, 2, 3, 4, 5, and 6 illustrates how a spreadsheet processes a continuous data stream according to the invention. Displayed is a spreadsheet program that computes the square root of the sum of the squares of the last three data elements of a continuous data stream. The spreadsheet program of Fig. 1 is constructed in 6 expressions entered in cells B1, B2, B3, B4, B5, and B6.

Cell B1 contains an external input function "INPUT(1)". The input function "INPUT(1)" accesses a data source "1" external to the spreadsheet. In this example, we assume that cell B1 updates periodically as a function of some unspecified external event, e.g., in response to data on the lines of a parallel port whenever a line is toggled, or in response to a pixel value at specified coordinates in an image at a particular time.

Cell B2 contains an instance of a delay line object, and is parameterized with four parameters: the input data stream (B1*B1), the clock source (B1), the number of elements in the shift register (3), and the clock divider (1).

Referring to Fig. 3, a delay line object 30 consists of a data buffer 32 configured as a variable length shift register (or as a delay line, FIFO, pipeline, etc.), and a method for clocking data through the buffer 34. The clock signal 34 indicates when data is valid for processing in the spreadsheet. Input data streams supply the clock signals 34, and "buffered interval" spreadsheet functions support the random access of elements and other analysis over the buffered interval. For example, parallel random access of data elements is performed using "tap functions" 36 that independently access single values from the delay line buffer 32 at particular specified locations by calling methods 37 of the delay line object. Analysis over the buffered interval includes any vector operation on the data buffer, including summing the buffer, and performing a standard deviation or other statistical analysis on the buffer. Further, other sources of clocking signals include a timer (e.g., a clock edge every 20 seconds), an image acquire signal, or a parallel port, or any external event, for example. Other spreadsheet entities

20

can also provide clock signals. The delay line object also includes a plurality of parameters 40, as detailed in Fig. 3A.

Referring to Fig. 3A, the delay line object is parameterized by four arguments. The first argument is the input data stream 31; the second argument is the clock source 33; the third argument is the number of elements in the shift register buffer (delay line) 35; and the forth argument is a clock divider 39.

In the example of Fig. 1, the data source is defined by an expression that squares the input data stream B1; the clock source is referenced to cell B1; the number of data elements in the shift register is three; and the sample rate is one per cycle. So, when data is received in B1, it is squared and input to a three-element delay line.

Cells B3, B4, and B5 access the delay line instantiated in B2 to return the values of all three elements in the delay line data buffer. The tap function TAP(B2,0) in cell B3 accesses the last data item to be input to the delay line in cell B2. The tap functions TAP(B2,1) and TAP(B2,2) in cells B4 and B5, respectively, access data that was input on the previous two clock cycles 1 and 2.

Cell B6 completes the processing by referencing B3, B4, and B5 in a computation of the square root of the sum of the three values B3, B4, and B5 returned from the delay line's buffer.

Fig. 2 shows the final four spreadsheet display states that result when the spreadsheet program of Fig. 1 processes the input data stream {1.00,2.00,3.00,4.00,5.00,6.00}:

At time (T-3) cells B3, B4 and B5 display the values 9.00, 4.00, and 1.00, respectively, resulting from the squaring of the first three input data elements 3.00, 2.00, and 1.00;

At time (T-2) the value 16.00, the square of the input data value B1=4.00 (B1*B1), shifts into the delay line;

At time (T-1) the value 25.00, the square of the input data value B1=5.00 (B1*B1) shifts into the delay line; and

At time (T-0) the value 36.00, the square of the input data value B1=6.00 (B1*B1) shifts into the delay line.

21

For every cycle, the square root of the sum of the delay line contents (Mag) is computed by the expression assigned to cell B6: (B6=SQRT(B3+B4+B5)).

The input function INPUT(1) receives data from an external source and updates the value assigned to B1. The delay line object in B2 (B2=DELAY(B1*B1,B1,3,1) is clocked by the input function, and a new value is shifted into the data buffer upon each new input. The three tap functions TAP(B2,0), TAP(B2,1),and TAP(B2,2) are dependent on the contents of the delay line B2, and update in an unspecified order. Finally, the expression in B6 (SQRT(B3+B4+B5)) is dependent on B3, B4, and B5, and is evaluated, the result being assigned to B6. At this point, all dependencies have been satisfied, and the contents of the spreadsheet remain unchanged until the next input event occurs.

In the forgoing example, the delay line expression in B2 behaves in a way very different from a conventional spreadsheet expression. First, the delay line is not a function; rather, it is an object instance, i.e., there is data storage (memory) allocated. Delay line class objects combine data storage with a member function that implements the shift register, for example.

Second, unlike a conventional spreadsheet expression that is evaluated only as necessary to update the spreadsheet, according to the invention, the evaluation of the delay line member function is executed only upon triggering by a clock signal. The value of the input to the delay line member function does not need to change to force the evaluation of the delay line member function. The state of the source data argument is irrelevant; the shift register action can only be activated by a signal from a valid clock source, such as the input function in the example.

Referring to Fig. 3, the delay line object 30 consists of a data buffer 32 configured as a variable length shift register (or as a delay line, FIFO, pipeline, etc.), and a method for clocking data through the buffer 34. The clock signal 36 indicates when data is valid for processing in the spreadsheet. Input data streams supply the clock signals 36, and buffered interval functions support the random access of elements, and other analysis over the buffered interval. For example, parallel random access of data elements is performed

22

using "tap functions" 38 that independently access single values from the delay line buffer 32 at particular specified locations by calling methods 37 of the delay line object. Analysis over the buffered interval includes any vector operation on the data buffer, including summing the buffer, and performing a standard deviation or other statistical analysis on the buffer. Further, other sources of clocking signals include a timer (e.g., every 20 seconds), an image acquire signal, a parallel port, or any external event, for example. Other spreadsheet entities can also provide clock signals.

The delay line object 30 also includes a plurality of parameters 40, as detailed in Fig. 3A, including, for example, the input data stream 31, the clock source 33, the number of elements in the shift register, and the clock divider 39.

Large Data Sets

Referring to Fig. 4, a preferred spreadsheet user interface for machine vision applications includes a monitor 41 connected by cable to a computer 42. The computer 42 includes image acquisition, processing, and graphical display capability. A camera and lens assembly 43 has an object surface 45 within its field of view, and is connected to the computer 42. Also, connected to the computer 42 is a user input device, such as a game controller 44, or a standard keyboard (not shown).

The screen of the monitor 41 depicts a semitransparent spreadsheet 46 superimposed on an image and graphics layer 47 to form a composite display of the invention. The transparency of the semitransparent spreadsheet 46 can be adjusted by the user. The semitransparent spreadsheet 46 can also be displayed in a split-screen arrangement wherein the image and graphics layer 47 is displayed on one portion of the screen of the monitor 41, and the semitransparent spreadsheet 46, which may be adjusted so as to no longer be transparent, can be displayed in another portion of the screen of the monitor 41. Alternatively, the image and graphics layer 47 can be displayed on the screen of the monitor 41, and the semitransparent spreadsheet 46, which may be adjusted so as to no longer be transparent, can be displayed on the screen of a second monitor (not shown).

23

Fig. 5 shows the superposition combination of the contents of data display buffer 51 and the semitransparent spreadsheet overlay 52 to form a composite display 53. Not shown are the split-screen combination, and the dual-screen combination of the contents of data display buffer 51 and the semitransparent spreadsheet overlay 52.

Figs. 6, 7, 8, and 9 illustrate the effect of spreadsheet cell selection on the underlying data display buffer. The contents of the underlying data display buffer changes in accordance with the particular cell that is currently selected. Also, the contents of the formula line changes in accordance with the particular cell that is currently selected. Note that only the data contents of the selected cell is displayed; all of the other data associated with the objects underlying the other cells remains invisible to the user. By contrast, all of the data in a standard spreadsheet is visible to a user, including a large standard spreadsheet, notwithstanding the need to scroll over the large spreadsheet.

In monitor 61 of Fig. 6, spreadsheet focus (highlight) 63, indicated by the frame border (or any selection highlighting), is on cell A1. Formula line 62 indicates that cell A1 is assigned the single method object ACQUIRE, which contains an image buffer resulting from its image acquisition method. The underlying data display buffer 64 shows the content of the image buffer associated with ACQUIRE.

In monitor 71 of Fig. 7, spreadsheet focus 73 is on cell A2, indicating that cell A2 is selected. Formula line 72 indicates that cell A2 is assigned the single method object PROCESS which contains the image buffer that results from the processing a rectangular region 75 in the image associated with cell A1. The other four arguments of the object PROCESS represent the coordinates of the upper-left corner, and the height and width, of the rectangular region 75 in the image associated with cell A1. The underlying data display buffer 74 shows the content of the image buffer associated with the processed image.

In monitor 81 of Fig. 8, spreadsheet focus 83 is on cell A3. Formula line 82 indicates that cell A3 in this example is assigned the single method object HISTEXTRACT which contains a histogram array extracted from a rectangular region 85 in the processed image assigned to cell A2. The other four arguments of the object PROCESS represent the coordinates of the

24

upper-left corner, and the height and width, of the rectangular region 85 in the image associated with cell A2. The underlying data display 84 shows the content of the input image buffer plus a graphical representation of the histogram array associated with the content of the input image buffer.

In monitor 91 of Fig. 9, spreadsheet focus 93 is on cell A4. Formula line 92 indicates that cell A4 in this example is assigned a contrast value computed from the histogram array assigned to cell A3. The underlying data display buffer 94 shows the content of the acquired image.

The forgoing illustrates how single method objects co-operate with a variably transparent spreadsheet overlaid on the contents of a data display buffer so as to efficiently and elegantly extend the conventional spreadsheet paradigm to accommodate images and other large data sets.

Conditional Cell Execution

Referring to Fig. 11, a sequence of four instances 101, 104, 7, 110 of a portion of an electronic spreadsheet incorporating the improvements of the invention is shown. In the first spreadsheet instance 101, cell B1 is selected by a user, and is seen to be equal to the logical constant FALSE. After a user selects a "cell state" display mode from a pull-down menu, selecting a cell or plurality of cells causes an enable dialog box 102 to appear as well as causing an associated logical switch expression pop-up window 103 to appear (also see Fig. 12, step 115). The enable dialog box 102, superimposed on the spreadsheet grid, shows that the enable switch expression is equal to the logical constant TRUE, indicating that the expression in B1 will be executed. It is also possible for the user to change the enable switch expression by entering a new enable switch expression 115, as shown in Fig. 12.

Thus, the value of B1 will be FALSE, provided that the enable switch expression that controls the execution of cell B1 is set to TRUE. The implicit conditional expression (including the enable switch expression) that controls the execution of cell B1 is shown in the pop-up window 103 that appears with

25

the enable dialog 102, and contains the logical expression:
IF(TRUE,B1=FALSE).

In spreadsheet instance 104, cell B2 is selected and conditionally set equal to the results of the expression evaluated in B3, based on the conditional expression in the pop-up window 106. Enable dialog 105 shows that the enable switch expression is equal to the result of expression evaluated in cell B1. Since B1 is known to evaluate to FALSE, in this case, the expression in B2 will not be executed. The implicit conditional expression associated with cell B2 is shown in the pop-up window 106.

In spreadsheet instance 107, cell B3 is selected and set equal to the result of an image measurement function performed on an external image data buffer. Enable dialog 108 shows that the enable switch expression is equal to the logical constant TRUE, indicating that the expression in B3 will be executed. The implicit conditional expression associated with cell B3 is shown in the pop-up window 109.

In spreadsheet instance 110, cell B4 is selected and set equal to the result of a logical operation performed on the results of the expressions evaluated in B2 and B3. Enable dialog 111 shows that the enable switch expression is equal to the logical constant TRUE, indicating that the expression in B4 will be executed. The implicit conditional expression associated with cell B4 is shown in the pop-up window 112.

Referring to Fig. 11, a BASIC program fragment 113 shows the BASIC program steps that together are equivalent to the electronic spreadsheet program shown in Fig. 10. One skilled in the art can observe that this BASIC program will exhibit the same behavior, i.e., evaluate expressions, assign values, and transfer program control, so as to achieve the same result as the execution of an improved electronic spreadsheet according to the invention, yet via an entirely different user interface. Fig. 11 emphasizes and illustrates that the improved electronic spreadsheet of the invention can achieve functionality that was previously not possible in electronic spreadsheets, and prior to the invention, was possible only using computer languages like BASIC, COBOL, FORTRAN, and C, for example.

In the forgoing description of Fig. 10, a spreadsheet is programmed to behave in one of two possible ways, depending on cell B1. If the expression in

26

cell B1 is set to TRUE (thereby indicating that the spreadsheet is in training mode in the example of Fig. 10), the spreadsheet executes a training sequence whereby the results of an image measurement executed in cell B3 are saved in cell B2 for future comparisons.

If the expression in cell B1 evaluates to FALSE (thereby indicating testing mode in the example of Fig. 10), the training step associated with the expression in cell B1 is not executed. The expression in cell B4 goes on to perform a test which compares the current image measurement performed in B3 to the measurement saved on the most recent training cycle in cell B2. Figure 13 depicts the spreadsheet main loop. At the start of processing, as indicated by the start bubble, the function calls the DisplaySheet function to display the contents of the spreadsheet. It then waits for one of six input conditions: Move Up, Move Down, Move Left, Move Right, Edit Cell, and Escape. It responds to any of the first four conditions by testing the current row or column count and incrementing or decrementing the respective one of these, all as indicated in Fig. 13. The program responds to an Edit Cell input condition by awaiting an input, testing the validity of the command, and if the comand is valid, then calling the Update CellCmd and EvaluateSheet function, as shown. The program responds to an Escape input by quitting.

Figure 14 depicts the DisplaySheet function. As shown here, the program initializes and increments the indices i and j in order to loop through the rows and columns, respectively, of the spreadsheet. For each combination of those indices within the spreadsheet, the program calls the Print Value function in order to display a cell at the location [i][j]. A test is performed during the inner loop to determine whether a location i and j points to the current row and column respectively and, if so, to highlight the cell (via the function Highlight Cell) prior to invoking the Print Value function.

Figure 15 depicts the EvaluateSheet function. At the outset, this function call the EvaluateCell function. Thereafter, as in Figure 14, the EvaluateSheet function initializes and increments the indices i and j in order to loop through the rows and columns, respectively, of the spreadsheet. The inner loop here, rather then invoking the aforementioned Print Value and Highlight Cell

functions, tests CellCmd[i][j] for dependency on CellVal[u][v]. If so, it calls the EvaluateSheet function. Otherwise, it proceeds to the next row and column, all as clearly shown in Fig. 15.

Figure 16 depicts the aforementioned EvaluateCell function of Fig. 15 which, at the onset, includes invoking the Split CellCmd[i][j] with the indicated command and arguments. Thereafter, the function evaluates arguments Arg1 -> Val1, Arg2 -> Val2, and so forth, as indicated. The function then executes one of the five respective commands: SET, ADD, MUL, SQRT, and DEL, as shown. The SET command results in the cell value being set to Val1, as shown. The Add command, results in the cell value being set to the sum of Val1 and Val2. Likewise, the MUL command results in the call value being set to the multiplicative product of Val1 and Val2. The SQRT command results in the cell value being set to the square root of Val1. The DEL command results in the cell value being set to zero and command being set to NULL. Following execution of the indicated command the function returns.

Spreadsheet with Extensions

Spreadsheet processing begins in the Main Loop, shown in Fig. 17. The Main Loop allocates storage and assigns variables to represent data storage of dimension N by M where, N is the number of rows in the spreadsheet, and M is the number of columns. The array element Formula[u][v] identifies storage for the spreadsheet formula assigned to the (u,v) coordinate of the grid. The array element Data[u][v] identifies storage for data associated with the output of Formula[u][v]. The local variables Row and Col identify the active cell. The active cell is the spreadsheet cell that has the current user interface focus and is the implicit target of editing and special display operations.

After entering at the start bubble, the DisplaySheet function is called to displays the contents of the spreadsheet. The program then waits for one of four input conditions: Move, Edit, Trigger, or Escape. The Move condition indicates that the user wants to change the coordinate of the active cell. The Update function updates the row and column coordinate of the active cell

28

within the spreadsheet grid. The Edit condition indicates that the user wants to modify the formula associated with the active cell. The EditCell function is called to assigns a new formula to the active cell. The ExecuteCell function is called to updates the data associated with the active cell and all dependent cells. The Trigger condition indicates that an external event has occurred. The ExecueTrigger function is called to locate and execute all cells within the spreadsheet dependent on that event.

After handling **Move**, **Edit**, or **Trigger** conditions the main processing loop is restarted. The **Escape** condition indicates that the user wants to terminate the main processing loop and exit the spreadsheet program.

The **DisplaySheet** function, shown in Fig. 18, is called from the **Main Loop** to displays the contents of the spreadsheet. Special emphasis given to information associated with the cell at focus coordinates **(u,v)**. On entering **DisplaySheet** the **DisplayData** function is called to handle special display requirements associated with any large data sets that may be assigned to the cell at the focus coordinates. The conventional spreadsheet grid is then displayed by looping through the cell array and, for each cell, calling the **DisplayCell** function. When the temporary grid coordinates **(i,j)** are equal to the focus coordinates **(u,v)** the function **HighLightCell** is called to highlight that position in the spreadsheet grid. Before return, the **DisplayFormula** function is called to display the formula assigned to the focus cell.

The **DisplayData** function, shown in Fig. 22, is called from **DisplaySheet** to manage the graphical display of large data sets contained within the cell at grid coordinate **(u,v)**. The function compares the cell formula with formulas that output large data sets. If a match is found the corresponding data display function is called. For example, if the cell has been configured to acquire an image, the **DisplayImage** function is called to display the acquired image data. Similarly, if the cell has been configured to compute a histogram, The **DisplayHist** function will be called to display the accumulated histogram.

The **DisplayCell** function, shown in Fig. 23, is called from **DisplaySheet** to control the representation of cell data in the conventional spreadsheet grid. The function compares the cell's formula to determine the type of data stored at **(u,v)**. If the data is a conventional scalar value, that

29

value is displayed at the cell's **(u,v)** grid location. If the cell contains extended data, the name of the stored data type is displayed cell's **(u,v)** grid location. For example, if the cell has been configured to acquire an image, the text string "IMAGE" is displayed. Similarly, if the cell has been configured to compute a histogram the text string "HIST" will be displayed.

The **EditCell** function, shown in Fig. 19, is called from the **Main Loop** to inputs formulas to be assigned it to the cell at coordinates **(u,v)**. The formula is first tested for valid syntax and argument type. If valid the **DeleteCell** function is called to release storage associated with the old formula. The old formula is then replaced by the new formula and the **AllocCell** function is called to allocate and initialize the necessary data structures.

The **AllocCell** function, shown in Fig. 28, is called from **EditCell** to manage the allocation and initialization data. The function compares the return type of the formula at **(u,v)** with extended data types. If a match is found the corresponding data allocation function is called and **Data[u][v]** is assigned a reference to the allocated storage. For example, if the cell's formula returns image data, the **AllocImage** function is called to allocate and initialize memory storage locations associated with image data. Similarly, if the formula is determined to return histogram data, the **AllocHist** function is called to allocate and initialize storage appropriate to the storage of histograms.

The **DeleteCell** function, shown in Fig. 29, is called from **EditCell** to manage the de-allocation of cell data. The function compares the return type of the formula at **(u,v)** with extended data types. If a match is found the corresponding data de-allocation function is called to release the storage referenced by **Data[u][v]**. For example, if the cell's formula returns image data, the **DeleteImage** function is called to de-allocate memory storage location associated with image data. Similarly, if the formula is determined to return histogram data, the **DeleteHist** function is called to de-allocate data storage associated with the array.

The **ExecuteCell** function, shown in Fig. 20, is called by the **Main Loop** and, recursively, by **ExecuteDependentCells**. The **ExecuteCell** function calls the **ExecutueFormula** function to update the contents of **Data[u][v]**, and the **ExecuteDependentCells** function to force the update of all cell dependent on **Data[u][v]**.

The **ExecuteFormula** Function, shown in Fig. 24, is called by **ExecuteCell** to manage the evaluation of the cell formula at spreadsheet grid coordinate **(u,v)**. The function views the formula in three parts: the switch expression, the command instructions, and the arguments. If the switch expression evaluates **TRUE**, the values of arguments are resolved, and a determination is made as to whether the cell is conventional or extended. If the cell contains a conventional spreadsheet formula returning a simple scalar the **ExecuteConventional** function is called to complete the evaluation of the cell's value. If the formula results in a large data set, such as an image or a histogram, the **ExecuteExtended** function is called to complete the evaluation of the cell's internal data structure.

The **ExecuteConventional** Function, shown in Fig. 26, is called from **ExecuteFormula** to manage the evaluation of conventional spreadsheet commands. The input command is compared with a list of valid commands, if a match is found the corresponding operation is executed updating the cells data value. For example, the conventional spreadsheet conditional command "IF" interprets the first argument as a condition by which to decide whether to set the cell's **Data[u][v]** to either the second or third argument. Similarly, the command "ADD" causes **Data[u][v]** to be set to the sum of the first and second arguments.

The **ExecuteExtended** Function, shown in Fig. 27, is called from **ExecuteFormula** to manage the evaluation of extended spreadsheet commands. The input command is compared with a list of valid commands, if a match is found the corresponding operation is executed updating the cells data value. For example, the "ACQUIRE" command invokes the **Acquire** function, which causes a new image data to be copied from an external source to the storage location identified by the cell's **Data[u][v]** variable. Similarly, the "HIST" commands invokes the **Histogram** function which causes a histogram of the image data identified by first argument to be

31

computed. The histogram array is stored in the locations identified by the cell's **Data[u][v]** variable.

The **ExecuteDependentCells** Function, shown in Fig. 25, is called by **ExecuteCell** and by **ExecuteTrigger**. The **ExecuteDependentCells** function locates and executes spreadsheet cell whose output is a function of the data stored in the cell located at spreadsheet grid coordinate **(u,v)**. This function searches the spreadsheet for cells whose input expressions reference the cell at **(u,v)**. If a match is found, a test is made to determine is the reference is circular, that is, if the formula is ultimately dependent on itself. If the reference is determined to be noncircular, the **ExecuteCell** function is called to execute the dependent cell. Note that the call to **ExecuteCell** is recursive, since **ExecuteCell** will call **ExecuteDependentCells** to evaluate subsequent dependencies.

The **ExecuteTrigger** function, shown in Fig. 21, is called from the **Main Loop** to search the spreadsheet for a trigger formula. If a matching trigger cell is found, its **Data[i][j]** variable is set **TRUE** and function **ExecuteDependentCells** is called to update any dependent cells. After all dependent cells have been executed the trigger cells value is set back to **FALSE**, to prevent the asynchronous execution of cells with explicit trigger dependencies.

Further understanding of the illustrated embodiments may be obtained by reference to the Appendix filed herewith, which discloses preferred methods and apparatuses for machine vision interfaces and environments.

Other modifications and implementations will occur to those skilled in the art without departing from the spirit and the scope of the invention as claimed. Accordingly, the above description is not intended to limit the invention except as indicated in the following claims.

CLAIMS

What is claimed is:

1. A user interface for programming machine vision applications, comprising:
   an image window for displaying an image to be analyzed;
   a spreadsheet for analyzing the image, displayed with the image window.

2. The user interface of claim 1, further comprising:
   a control pad, at least for navigating over the spreadsheet.

3. The user interface of claim 1, further comprising:
   a plurality of menus having various software tools.

4. The user interface of claim 1, further comprising:
   a palate of alpha-numeric characters for input into the spreadsheet.

5. The user interface of claim 1, wherein the spreadsheet is semi-transparent, such that the image can be seen through the spreadsheet.

6. The user interface of claim 1, wherein the spreadsheet is adapted to perform conditional cell execution.

7. The user interface of claim 1, wherein the spreadsheet includes functions that can operate on values stored over an interval of time.

8. The user interface of claim 2, wherein the control pad indicates a particular cell of the spreadsheet, thereby resulting in a change in the appearance of the image displayed.

9. The user interface of claim 1, wherein the spreadsheet includes cells adapted to buffer outputs, thereby providing circular reference for recursive operation.

10. The user interface of claim 1, wherein the spreadsheet includes cells that can be assigned arbitrary depth in a time dimension.

11. The user interface of claim 1, wherein complex vision objects are instantiated as a single cell in the spreadsheet, the single cell having data that is accessible through a plurality of member functions.

12. The user interface of claim 1, wherein the spreadsheet includes cell that are adapted to store two dimensional image arrays that result from image processing operations.

13. The user interface of claim 1, wherein the spreadsheet includes image buffers used for intermediate processing, the image buffers being automatically released once dependent operation are satisfied.

14. The user interface of claim 1, further comprising:
a spreadsheet mask for obscuring portions of the spreadsheet, leaving portions unobscured that can be used to control operation of the spreadsheet.

15. An electronic spreadsheet for processing a continuous data stream, the electronic spreadsheet having a plurality of cells, the improvement comprising:

a clocked delay line object, the clocked delay line object being instantiated and assigned to a cell of the electronic spreadsheet;

an external data input function adapted to provide a clock signal to the clocked delay line object; and

a spreadsheet function adapted to provide random access to data stored in the clocked delay line object.

16. The electronic spreadsheet of claim 15, wherein the clocked delay line object is parameterized by a plurality of arguments, including:

an input data stream;

a clock source;

a number of elements in a shift register buffer; and

a clock divider.

17. The electronic spreadsheet of claim 15, wherein the clocked delay line object further includes:

a method for clocking data; and

a method for access to data stored in the clocked delay line object.

18. The electronic spreadsheet of claim 15, wherein the clocked delay line object is adapted to trigger the execution of the shift register function of the delay line object upon receipt of a clock signal provided by the external data input function.

19. The electronic spreadsheet of claim 15, further including a spreadsheet function that performs analysis of a data interval stored in the clocked delay line object.

20. A method for using an electronic spreadsheet for processing continuous data streams over an interval of size greater than one, the method comprising:

instantiate and assign a clockable delay line object having

data storage to at least one cell of the electronic spreadsheet;

provide a clock signal to the delay line object;

randomly access the data storage over an interval of size

greater than one; and

process the data so-accessed.

21. A method for processing continuous data streams over an interval of size

greater than one using an electronic spreadsheet having clocked delay line

objects instantiated in the cells of the electronic spreadsheet, each delay line

object having a delay line buffer, a source of clock signals, and spreadsheet

functions that support selectable access to data stored in a delay line of the

clocked delay line object, the method comprising:

assign to a first cell an external input function adapted to

access a data source external to the spreadsheet;

assign to a second cell an instance of a delay line object;

assign to a plurality of cells a respective plurality of tap

functions, each tap function adapted to access the delay line

buffer of the delay line object at a selectable clock cycle to provide

a respective plurality of data elements stored in the delay line

buffer; and

assign to at least one cell a function adapted to process the

plurality of data elements upon each clock cycle.

22. An electronic spreadsheet having a plurality of cells, the improvement comprising:

a single method object, adapted to be instantiated in at least one of the spreadsheet cells, and adapted to provide internal storage and member functions, the member functions being adapted to access the single method object and return a single value;

a data display buffer, the data contents of which are displayed under a variably transparent spreadsheet grid; and

means for selectively displaying the data content of the single method object in the data display buffer corresponding to a selected spreadsheet cell.

23. A method for selectively displaying large data sets in an electronic spreadsheet having a plurality of cells, the method comprising:

instantiating a single method object in each of a plurality of the cells of the spreadsheet, each single method object being adapted to provide internal storage for storing a large data set;

displaying the large data set of the single method object corresponding to a selected cell of the spreadsheet; and

displaying in superimposed relationship with the large data set a variably transparent spreadsheet including the selected cell.

24. A user-interface method for selectively displaying machine vision images stored in an electronic spreadsheet having a plurality of cells, the method comprising:

37

instantiating a single method object in each of a plurality of the
cells of the spreadsheet, each single method object being adapted to
provide internal storage for storing a machine vision image;

selecting a cell from the plurality of cells;

displaying the machine vision image stored in the single method
object corresponding to the selected cell; and

displaying in superimposed relationship with the machine vision
image a transparent electronic spreadsheet including the selected
cell.

25. The user-interface method of claim 24, wherein the transparent electronic
spreadsheet is adjustably transparent.

26. The user-interface method of claim 24, wherein the selected cell is
selected using a hand-held controller.

27. The user-interface method of claim 24, wherein the selected cell is
selected using one of a standard keyboard and a mouse.

28. The user-interface method of claim 24, wherein the machine vision image
includes a superposition of an object image, and a graphical representation of
an analysis of the object image.

29. The user-interface method of claim 28, wherein the analysis of the object
image is a histogram of the object image.

30. An improved electronic spreadsheet having a plurality of cells, each cell having an executable expression, the improvement comprising:

    for each cell of said plurality of cells, a cooperative implicit conditional execution wrapper having a logical switch expression, the implicit conditional execution wrapper combining the executable expression with the logical switch expression so as to enable or disable the executable expression of the cell in accordance with the value of the logical switch expression.

31. The electronic spreadsheet of claim 30, further comprising;

    a user interface for selecting a cell or range of cells.

32. The electronic spreadsheet of claim 30, further comprising:

    a user interface for assigning a logical switch expression to an implicit conditional execution wrapper.

33. The electronic spreadsheet of claim 30, wherein the logical switch expression is a legal spreadsheet expression that can be evaluated to logical TRUE or logical FALSE.

34. The electronic spreadsheet of claim 33, wherein if the logical switch expression evaluates TRUE, the cell's corresponding expression is evaluated, a new result value is returned, and the cell's internal result buffer is updated.

35. The electronic spreadsheet of claim 33, wherein if the logical switch expression evaluates FALSE, the cell's expression is not evaluated, and the cell's result buffer is returned unmodified.

36. The electronic spreadsheet of claim 33, wherein individual switch expressions are initialized to a constant logical TRUE.

37. The electronic spreadsheet of claim 31, wherein the user interface includes an interactive assignor that interactively assigns a switch expression to any cell or range of cells within the spreadsheet.

38. The electronic spreadsheet of claim 30, wherein the user interface includes a cell selector that selects an arbitrary set of cells, and an interactive assignor that selectively assigns a logical switch expression to each set of cells so-selected, so as to program a desired behavior.

39. A method for programming conditional execution of program statements in an electronic spreadsheet having a plurality of cells, the method comprising:

associating an implicit conditional execution wrapper with each cell;

selecting at least one cell; and

entering a logical switch expression into the implicit conditional execution wrapper associated with the selected at least one cell.

40

40. An apparatus for programming conditional execution of program statements in an electronic spreadsheet having a plurality of cells, the method comprising:

means for associating an implicit conditional execution wrapper with each cell;

means for selecting at least one cell; and

means for entering a logical switch expression into the implicit conditional execution wrapper associated with the selected at least one cell.

B1 = INPUT(1) 1

		A	B	C	D	E
1		Data1:	3.00			
2		DLine:	DELAY			

B2 = DELAY(B1*B1,B1,3,1) 2

		A	B	C	D	E
1		Data1:	3.00			
2		DLine:	DELAY			

B3 = TAP(B2,0) 3

		A	B	C	D	E
1		Data1:	3.00			

B4 = TAP(B2,1) 4

		A	B	C	D	E
1		Data1:	3.00			

B5 = TAP(B2,2) 5

		A	B	C	D	E
1		Data1:	3.00			

B6 = SQRT(B3+B4+B5) 6

		A	B	C	D	E
1		Data1:	3.00			
2		DLine:	DELAY			
3		Tap 1:	9.00			
4		Tap 2:	4.00			
5		Tap 3:	1.00			
6		Mag:	3.742			

Fig. 1

B6 = SQRT(B3+B4+B5)

1	Data1:	3.00		
2	DLine:	DELAY		
3	Tap 1:	9.00		
4	Tap 2:	4.00		
5	Tap 3:	1.00		
6	Mag:	2.742		

Time = T - 3

B6 = SQRT(B3+B4+B5)

1	Data1:	4.00		
2	DLine:	DELAY		
3	Tap 1:	16.00		
4	Tap 2:	9.00		
5	Tap 3:	4.00		
6	Mag:	5.385		

Time = T - 2

B6 = SQRT(B3+B4+B5)

1	Data1:	5.00		
2	DLine:	DELAY		
3	Tap 1:	25.00		
4	Tap 2:	16.00		
5	Tap 3:	9.00		
6	Mag:	7.071		

Time = T - 1

B6 = SQRT(B3+B4+B5)

1	Data1:	6.00		
2	DLine:	DELAY		
3	Tap 1:	36.00		
4	Tap 2:	25.00		
5	Tap 3:	16.00		
6	Mag:	8.775		

Time = T - 0

Fig. 2

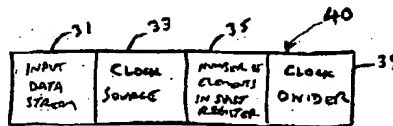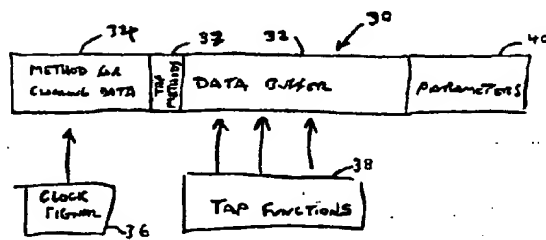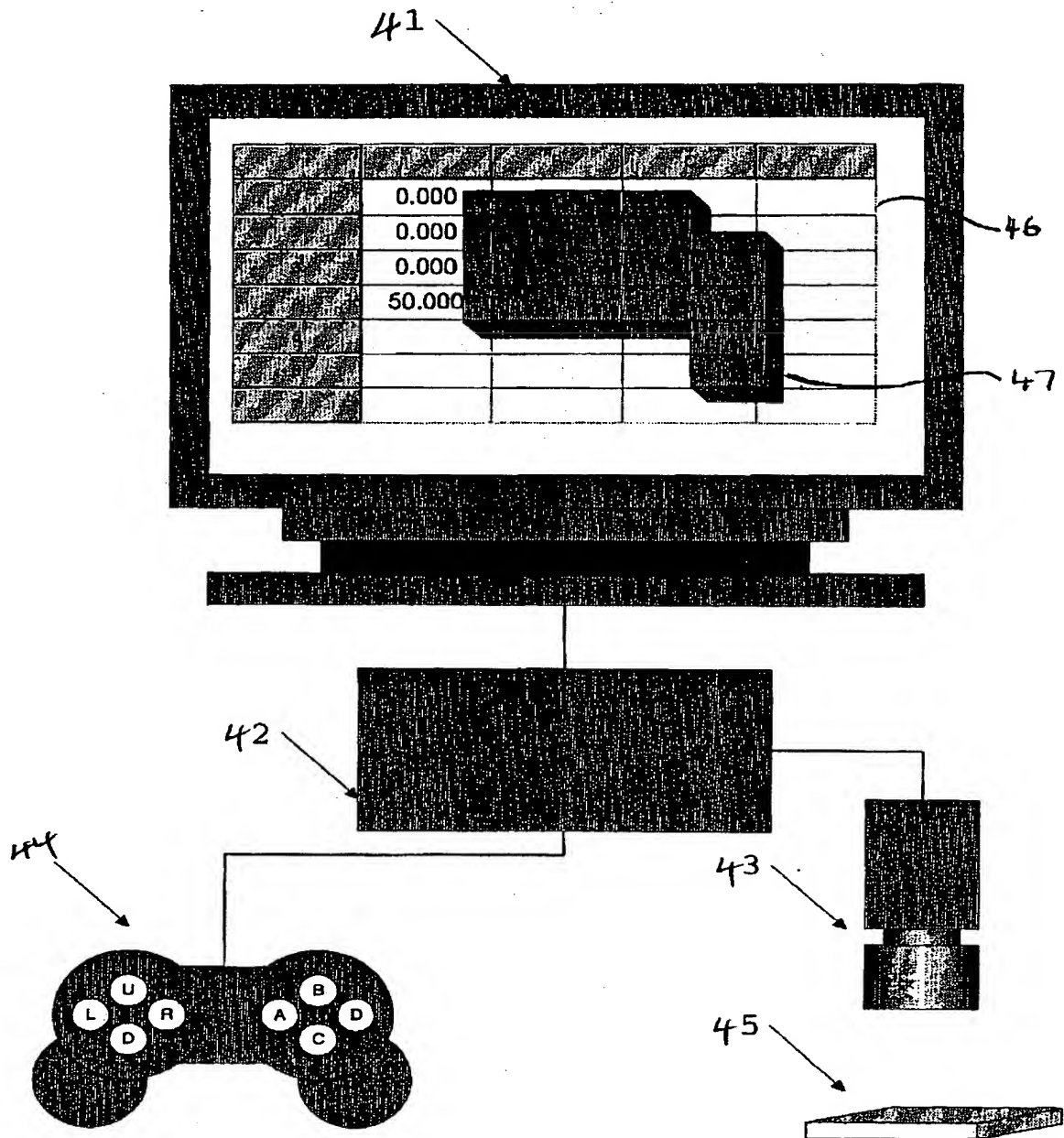| INPUT DATA STREAM	CLOCK SOURCE	NUMBER OF CLOCKS IN SHIFT REGISTER	CLOCK DIVIDER

FIG. 3A

FIG. 3

FIG. 4

FIG. 5

61
62
63
64

`A1 = ACQUIRE(0)`

```
0.000
0.000
0.000
50.000
```

FIG. 6

81
82
83
74

`A2 = PROCESS(A1,150,150,200,300)`

```
0.000
0.000
0.000
50.000
```

FIG. 7

81

82

```
A3 = HISTEXTRACT(A2,150,150,200,300)
```

83

84

	0.000				
	0.000				
	0.000				
	50.000				

FIG. 8

91

92

94

```
A4 = HISTCONTRAST(A3)*100/256
```

93

	0.000				
	0.000				
	0.000				
	50.000				

FIG. 9

5086503329

B1 = FALSE

		A	B	C
1		Train:	FALSE	
2		Dist 1:	23.000	
3		Dist 2:	24.000	
4		Test:	TRUE	

TRUE

OK

Cancel

102

103 IF(TRUE,B1= FALSE)

B2 = B3

		A	B	C
1		Train:	FALSE	
2		Dist 1:	23.000	
3		Dist 2:	24.000	
4		Test:	TRUE	

B1

OK

Cancel

104

105

106 IF(B1,B2=B3)

B3 = Measure(Image)

		A	B	C
1		Train:	FALSE	
2		Dist 1:	23.000	
3		Dist 2:	24.000	
4		Test:	TRUE	
5				

TRUE

OK

Cancel

107

108

109 IF(TRUE,B3=Measure(Image))

B4 = (B3>B2-2)&(B3<B2+2)

		A	B	C
1		Train:	FALSE	
2		Dist 1:	23.000	
3		Dist 2:	24.000	
4		Test:	TRUE	
5				
6				

TRUE

IF(TRUE,B4=(B3>B2-2)&(B3<B2+2))

Cancel

110

111

112
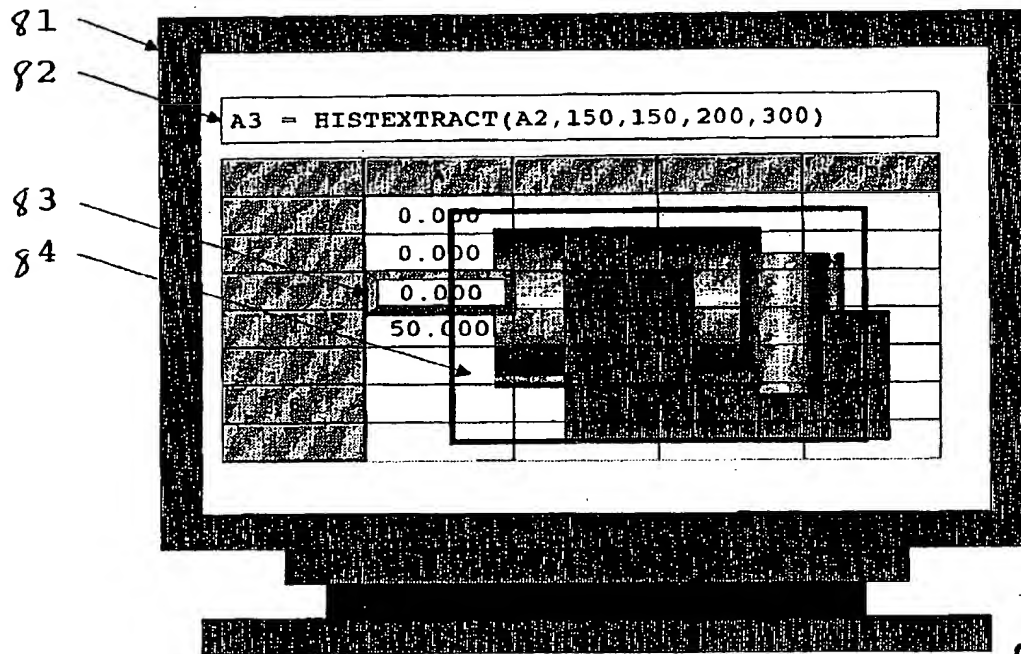
FIG. 10

```
10 B3=Measure(Image)
20 IF(B1=FALSE) GOTO 40
30 B2 = B3
40 B4=(B3>B2-2)&(B3<B2+2)
```

113

FIG. 11

ASSOCIATE IMPLICIT
CONDITIONAL EXECUTION
WRAPPER WITH EACH CELL — 114

SELECT AT LEAST
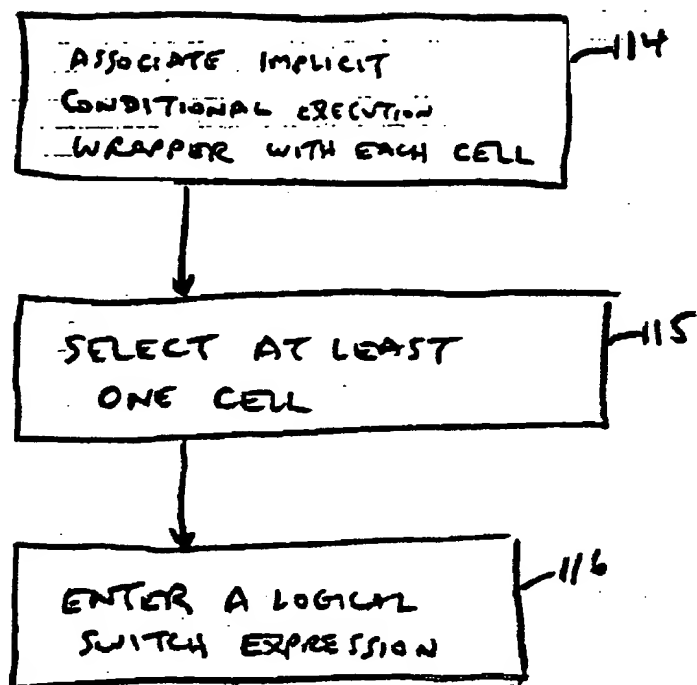ONE CELL — 115

ENTER A LOGICAL
SWITCH EXPRESSION — 116

FIG.12

Spreadsheet Main Loop
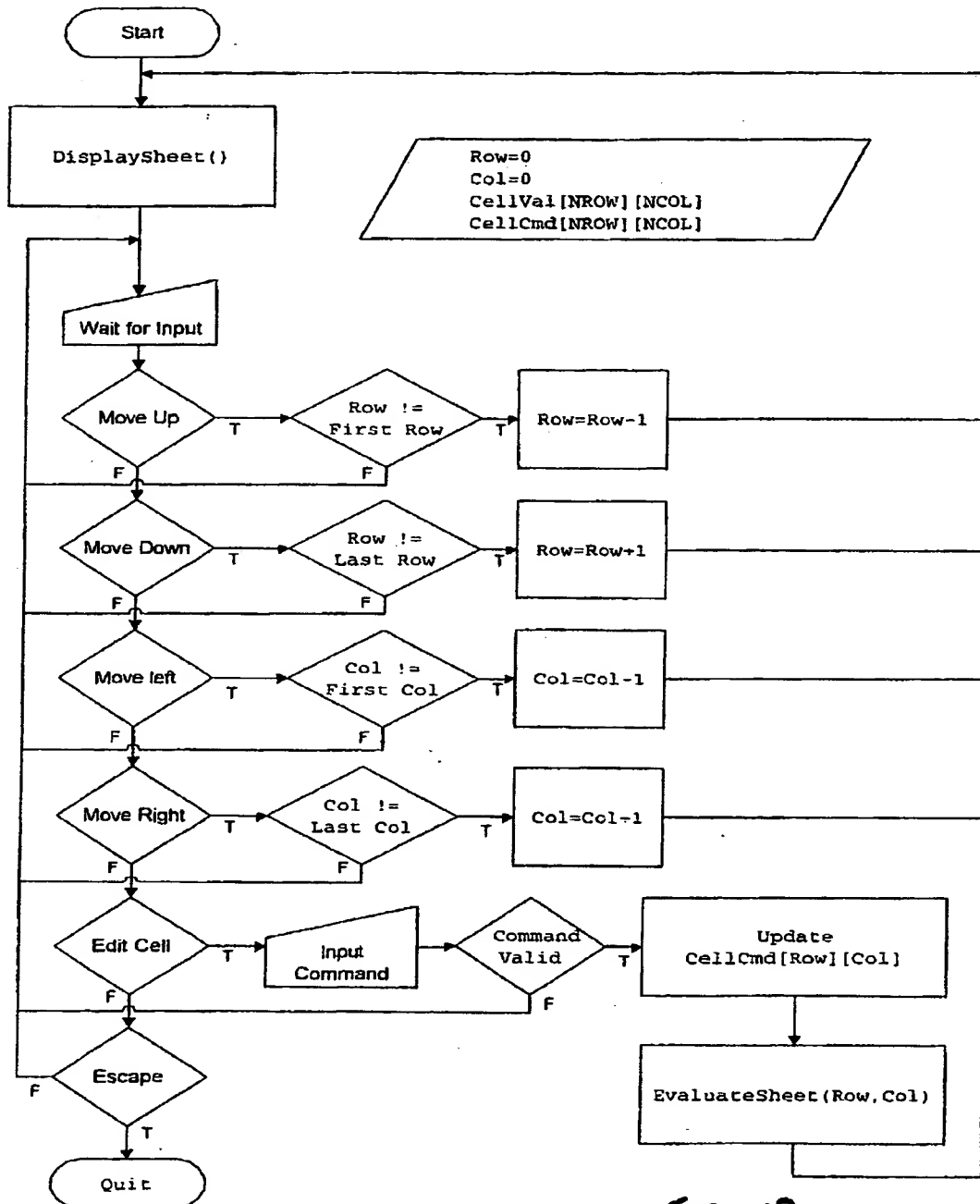
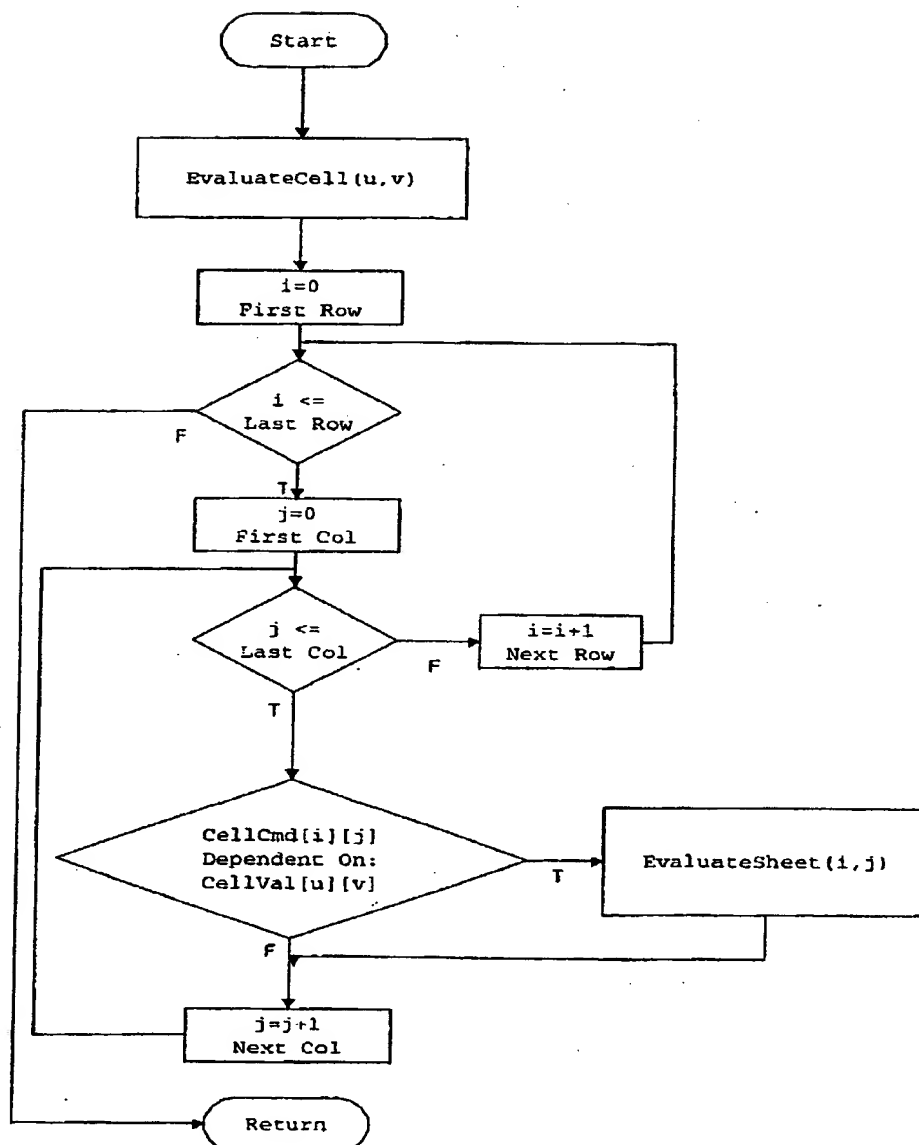
FIG. 13

DisplaySheet()

FIG. 14

EvaluateSheet(u,v)

```
┌─────────────┐
│    Start    │
└─────────────┘
       │
       ▼
┌──────────────────────┐
│  EvaluateCell(u,v)   │
└──────────────────────┘
       │
       ▼
┌──────────────────────┐
│        i=0           │
│     First Row        │
└──────────────────────┘
       │
       ▼
      ╱╲
     ╱  ╲
    ╱ i <= ╲      F
   ╱ Last Row ╲────────
    ╲        ╱
     ╲      ╱
      ╲    ╱
       ╲  ╱
        ╲╱
       T │
         ▼
┌──────────────────────┐
│        j=0           │
│     First Col        │
└──────────────────────┘
         │
         ▼
        ╱╲
       ╱  ╲
      ╱ j <= ╲        F    ┌──────────────┐
     ╱ Last Col ╲──────────│   i=i+1      │
      ╲        ╱           │  Next Row    │
       ╲      ╱            └──────────────┘
        ╲    ╱
         ╲  ╱
          ╲╱
         T │
           ▼
          ╱──────────╲
         ╱  CellCmd[i][j] ╲      T    ┌──────────────────────┐
        ╱  Dependent On:   ╲──────────│  EvaluateSheet(i,j)  │
        ╲  CellVal[u][v]   ╱          └──────────────────────┘
         ╲                ╱
          ╲──────────────╱
         F │
           ▼
┌──────────────────────┐
│       j=j+1          │
│     Next Col         │
└──────────────────────┘
           │
           ▼
    ┌─────────────┐
    │   Return    │
    └─────────────┘
```

F1G. 15

EvaluateCell(i,j)


```
                    ( Start )
                        │
                        ▼
        ┌───────────────────────────┐
        │ Split CellCmd[i][j]        │
        │ Cmd  - Command             │
        │ ArgX - Arguments           │
        └───────────────────────────┘
                        │
                        ▼
        ┌───────────────────────────┐
        │ Evaluate Arguments:        │
        │    Arg1 -> Val1            │
        │    Arg2 -> Val2            │
        │       ---                  │
        │    ArgN -> ValN           │
        └───────────────────────────┘
                        │
                        ▼
                   ╱ Cmd == ╲        ┌────────────────────────────┐
                  ╱  "SET"   ╲──────▶│  CellVal[i][j] = Val1      │───┐
                   ╲        ╱        └────────────────────────────┘   │
                    ╲      ╱                                          │
                        │                                             │
                        ▼                                             │
                   ╱ Cmd == ╲        ┌────────────────────────────┐   │
                  ╱  "ADD"   ╲──────▶│  CellVal[i][j] = Val1 +    │───┤
                   ╲        ╱        │           Val2             │   │
                    ╲      ╱         └────────────────────────────┘   │
                        │                                             │
                        ▼                                             │
                   ╱ Cmd == ╲        ┌────────────────────────────┐   │
                  ╱  "MUL"   ╲──────▶│  CellVal[i][j] = Val1 *    │───┤
                   ╲        ╱        │           Val2             │   │
                    ╲      ╱         └────────────────────────────┘   │
                        │                                             │
                        ▼                                             │
                   ╱ Cmd == ╲        ┌────────────────────────────┐   │
                  ╱  "SQRT"  ╲──────▶│  CellVal[i][j] = SQRT(Val1)│───┤
                   ╲        ╱        └────────────────────────────┘   │
                    ╲      ╱                                          │
                        │                                             │
                        ▼                                             │
                   ╱ Cmd == ╲        ┌────────────────────────────┐   │
                  ╱  "DEL"   ╲──────▶│  CellVal[i][j] = 0         │───┤
                   ╲        ╱        │  CellCmd[i][j] = "NULL"    │   │
                    ╲      ╱         └────────────────────────────┘   │
                        │                                             │
                        ▼                                             │
                    ( Return ) ◀─────────────────────────────────────┘
```

FIG. 16

12/25

Main Loop

```
Global Variables:
Formula[N][M]        // Cell Formula
Data[N][M]           // Cell Data Storage

Local Variables:
Row    // Cell Row Coordinate
Col    // Cell Column Coordinate
```

Start

DisplaySheet(Row,Col)

Input

Move —Yes→ Update(Row,Col)

No

Edit —Yes→ EditCell(Row,Col) → ExecuteCell(Row,Col)

No

Trigger —Yes→ ExecuteTrigger()

No

Escape —Yes→ Exit

No

Fig. 17

DisplaySheet(u,v)

FIG. 18

EditCell(u,v)


```
            ┌─────────┐
            │  Enter  │
            └────┬────┘
                 │
                 ▼
           ┌──────────┐
           │  Input:  │
           │NewFormula│
           └────┬─────┘
                │
                ▼
         ◇─────────────────◇
         Valid(NewFormula) ─────────┐
         ◇─────────────────◇        │
                 │ Yes              │
                 ▼                  │
         ┌────────────────┐         │
         │ DeleteCell(u,v)│         │ No
         └───────┬────────┘         │
                 │                  │
                 ▼                  │
     ┌────────────────────────┐     │
     │Formula[u][v]=NewFormula │    │
     └───────────┬────────────┘     │
                 │                  │
                 ▼                  │
         ┌────────────────┐         │
         │  AllocCell(u,v)│         │
         └───────┬────────┘         │
                 │                  │
                 ▼                  │
            ┌─────────┐             │
            │ Return  │◄────────────┘
            └─────────┘
```

FIG. 19

ExecuteCell(u,v)

```
              ┌─────────┐
              │  Enter  │
              └────┬────┘
                   │
                   ▼
        ┌──────────────────────────┐
        │   ExecuteFormula(u,v)     │
        └────────────┬─────────────┘
                     │
                     ▼
        ┌──────────────────────────┐
        │ ExecuteDependentCells(u,v)│
        └────────────┬─────────────┘
                     │
                     ▼
              ┌─────────┐
              │ Return  │
              └─────────┘
```

FIG. 20

ExecuteTrigger(u,v)
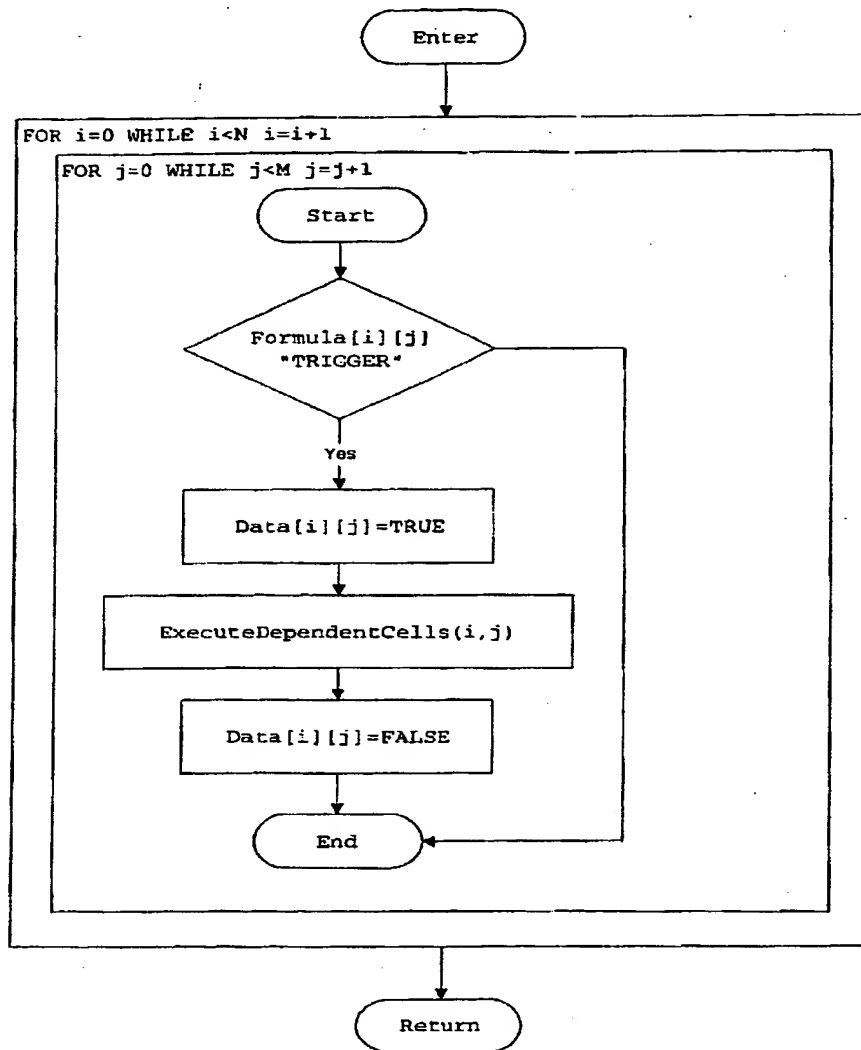
FIG. 21

DisplayData(u,v)

FIG. 22

DisplayCell(u,v)

```
                    ┌─────────────┐
                    │    Enter    │
                    └─────────────┘
                           │
                           ▼
                      ╱ "TRIGGER" ╲ ─────────→  ┌──────────────────────┐
                      ╲           ╱             │  Display("TRIGGER")  │──┐
                           │                    └──────────────────────┘  │
                           ▼                                               │
                      ╱  "FILTER"  ╲ ────────→  ┌──────────────────────┐  │
                      ╲            ╱             │  Display("IMAGE")    │─→│
                           │                    └──────────────────────┘  │
                           ▼                                               │
                      ╱ "ACQUIRE"  ╲ ────────→  ┌──────────────────────┐  │
                      ╲            ╱             │  Display("IMAGE")    │─→│
                           │                    └──────────────────────┘  │
                           ▼                                               │
                      ╱   "HIST"   ╲ ────────→  ┌──────────────────────┐  │
                      ╲            ╱             │  Display("HIST")     │─→│
                           │                    └──────────────────────┘  │
                           ▼                                               │
                      ╱  "SHIFT"   ╲ ────────→  ┌──────────────────────┐  │
                      ╲            ╱             │  Display("SHIFT")    │─→│
                           │                    └──────────────────────┘  │
                          No                                              │
                           ▼                                              │
              ┌──────────────────────┐                                   │
              │ Display(Data[u][v])  │                                   │
              └──────────────────────┘                                   │
                           │                                             │
                           ▼                                             │
                    ┌─────────────┐                                     │
                    │   Return    │←────────────────────────────────────┘
                    └─────────────┘
```

FG. 23

ExecuteFormula(i,j)

```
                    ┌─────────────┐
                    (    Start    )
                    └──────┬──────┘
                           │
        ┌──────────────────┴──────────────────┐
        │ Split Formula[i][j]:                 │
        │    Switch  - Condition               │
        │    Cmd     - Command                 │
        │    ArgX    - Arguments               │
        └──────────────────┬──────────────────┘
                           │
                      ╱─────────╲
                     ╱  Switch   ╲───────────────────────────────────┐
                     ╲  TRUE      ╱                                    │
                      ╲─────────╱                                     │
                         │Yes                                         │
        ┌────────────────┴────────────────┐                          │
        │ Evaluate Arguments:              │                          │
        │    Arg1 -> Val1                  │                          │
        │    Arg2 -> Val2                  │                          │
        │       ---                        │                          │
        │    ArgN -> ValN                  │                         No│
        └────────────────┬────────────────┘                          │
                         │                                            │
                    ╱─────────╲                                       │
                   ╱Conventional╲──Yes──┌─────────────────────────┐  │
                   ╲  Formula?   ╱       │  ExecuteConventional()  │──┤
                    ╲─────────╱          └─────────────────────────┘  │
                         │                                            │
                    ╱─────────╲                                       │
                   ╱  Extended  ╲──Yes──┌─────────────────────────┐  │
                   ╲  Formula?   ╱       │   ExecuteExtended()     │──┤
                    ╲─────────╱          └─────────────────────────┘  │
                         │                                            │
                        No──────┌─────────────┐◄─────────────────────┘
                                (    Return   )
                                └─────────────┘
```
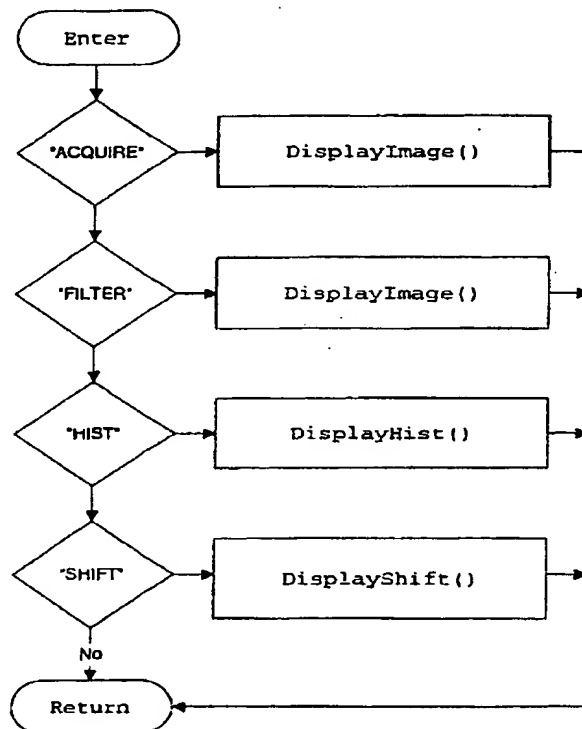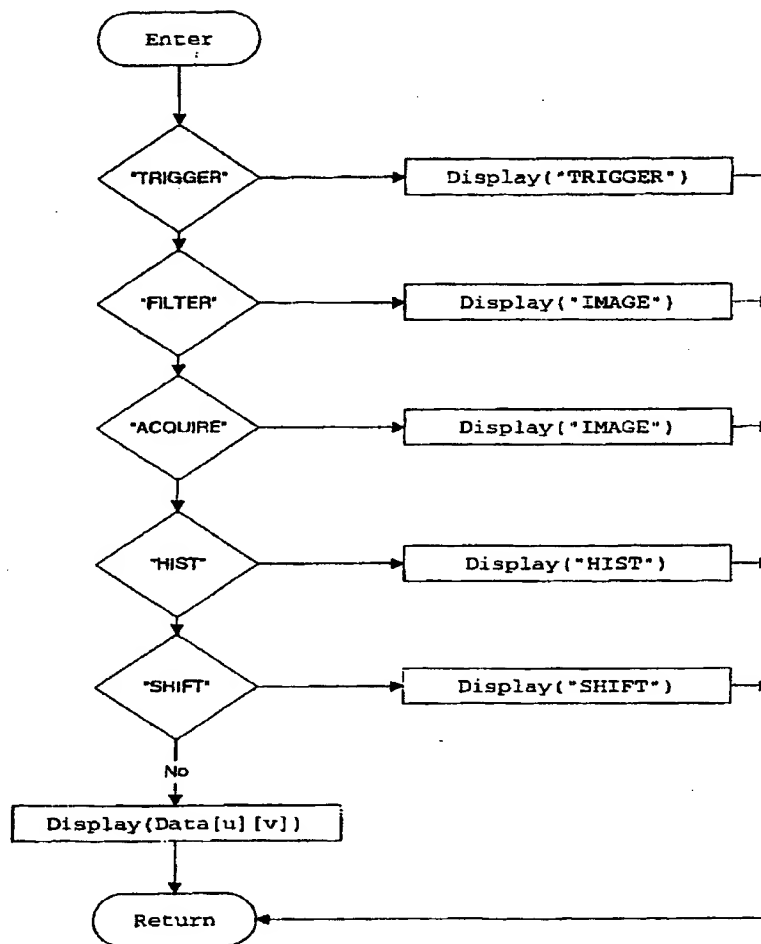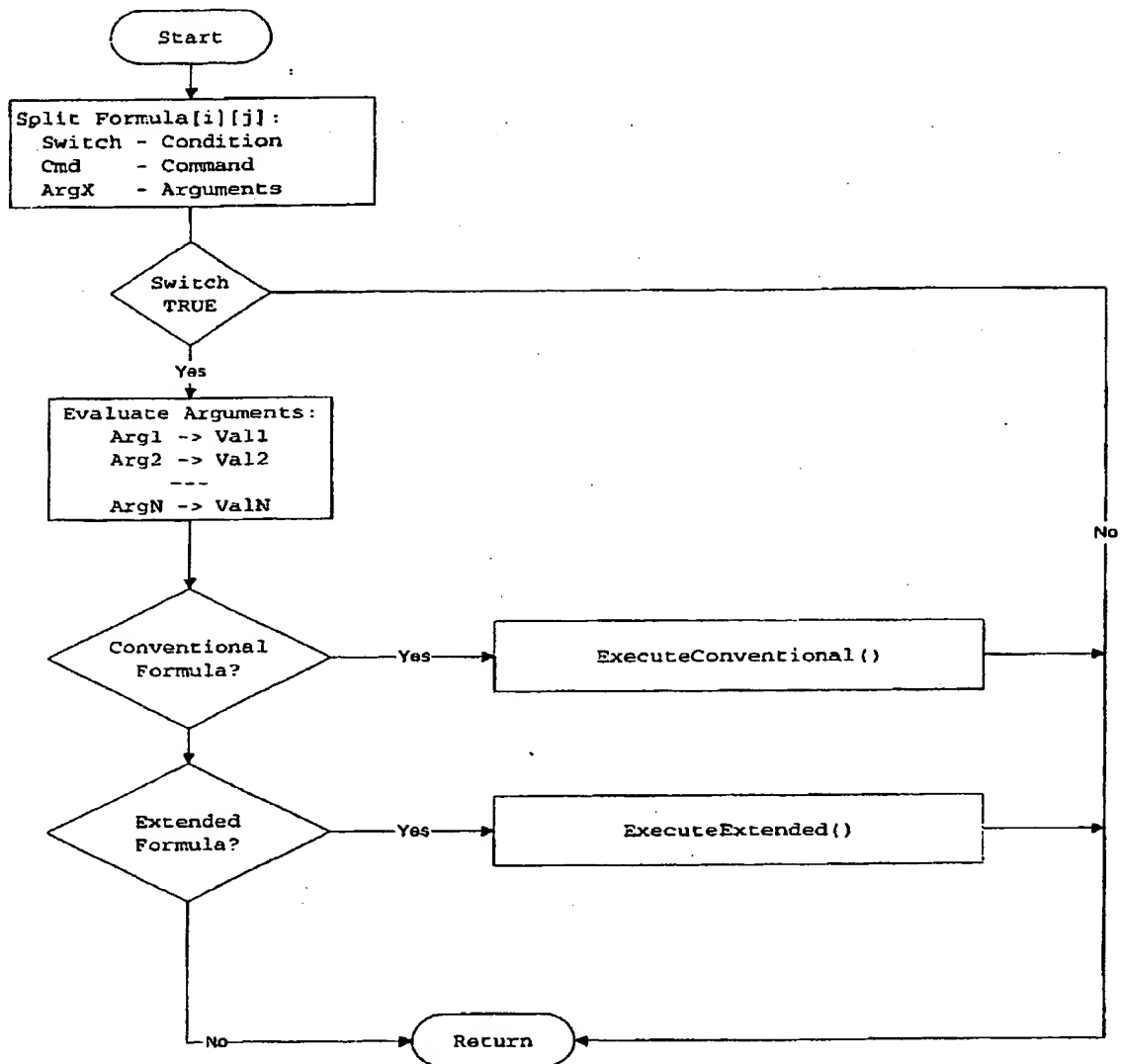
FIG. 24

ExecuteDependentCells(u,v)

FIG. 25

ExecuteConventional()

Enter

"IF" → Val1 TRUE → Yes → Data[i][j] = Val2
       → No → Data[i][j] = Val3

"SET" → Data[i][j] = Val1

"ADD" → Data[i][j] = Val1 + Val2

"SUB" → Data[i][j] = Val1 - Val2

"MUL" → Data[i][j] = Val1 * Val2

"DIV" → Data[i][j] = Val1 / Val2

"SQRT" → Data[i][j] = SQRT(Val1)

No

Return
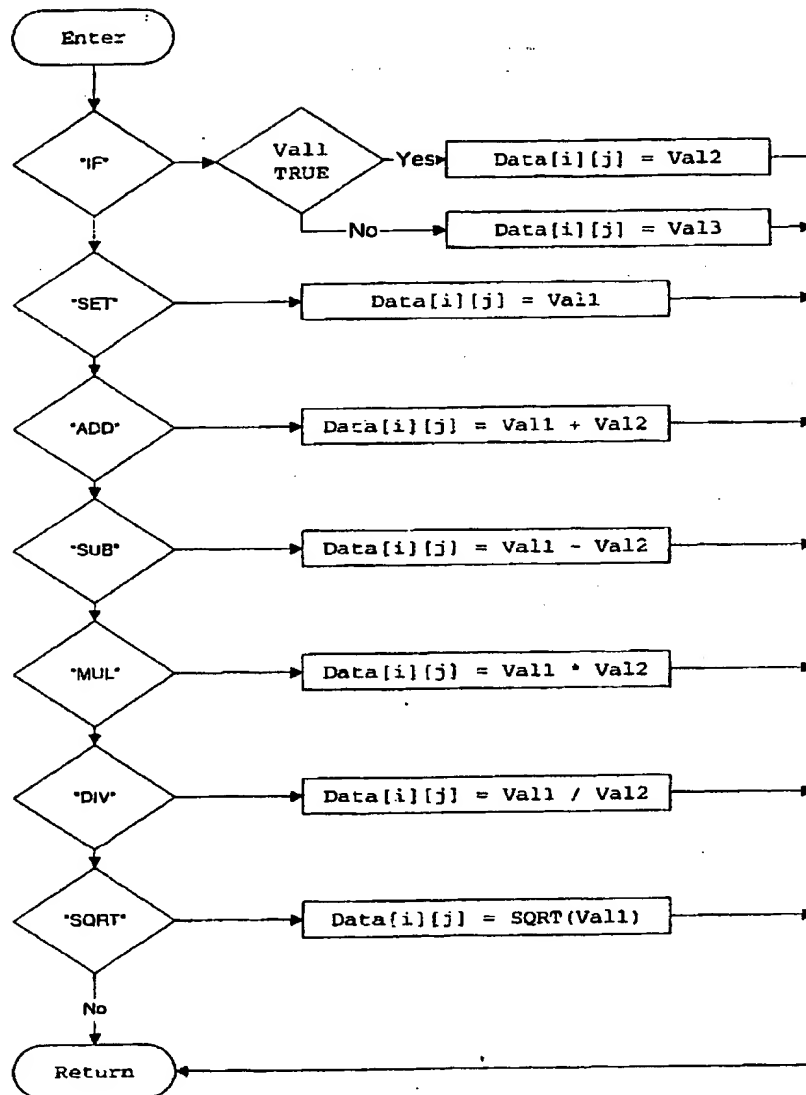
FIG. 26

ExecuteExtended()

FIG. 27

AllocCell(u,v)

FG. 28

DeleteCell(u,v)

```
        ( Enter )
            |
            v
        /ACQUIRE/ ----> | DeleteImage(u,v) | ----+
            |                                     |
            v                                     |
        /FILTER/ ------> | DeleteImage(u,v) | --->|
            |                                     |
            v                                     |
        /HIST/ --------> | DeleteHist(u,v) | ---->|
            |                                     |
            v                                     |
        /SHIFT/ -------> | DeleteShift(u,v) | --->|
            |                                     |
           No                                     |
            v                                     |
        ( Return ) <-----------------------------+
```
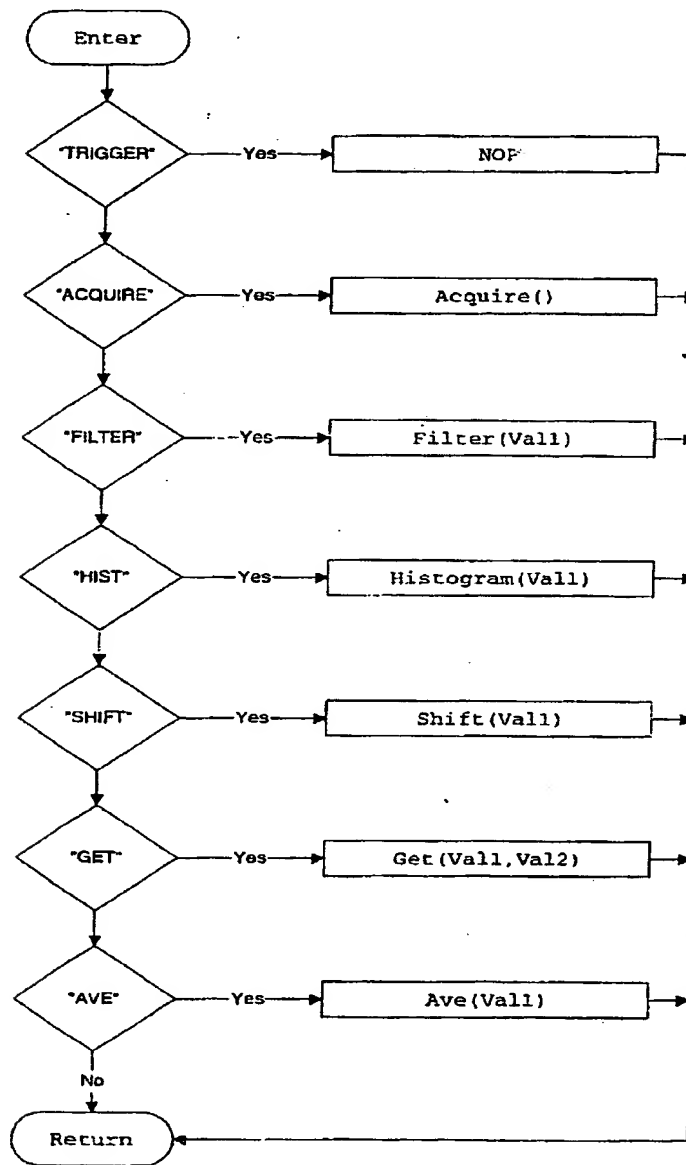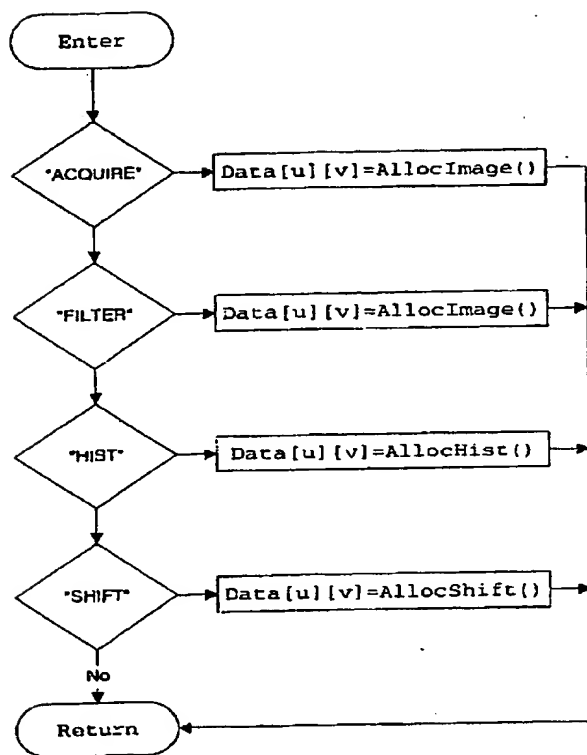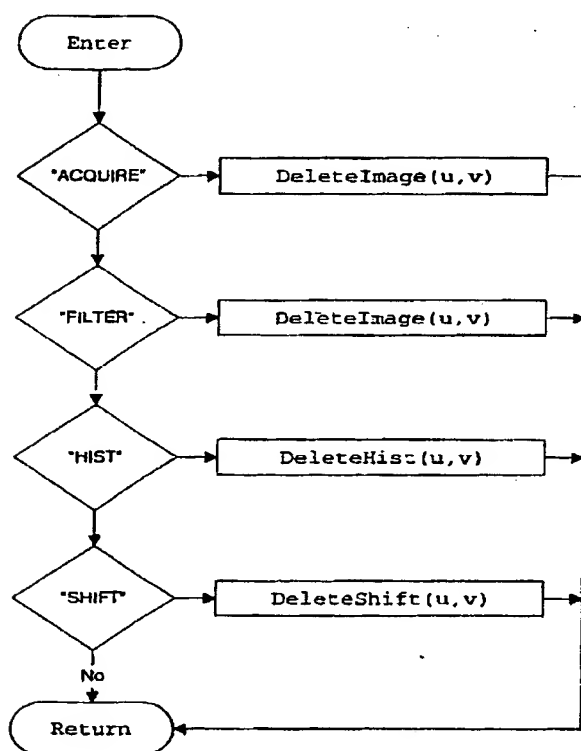
FIG. 29

◄ ► In-Sight Guide and Reference

In-Sight version 1.0
Copyright © 1999 by Cognex Corporation
www.cognex.com

Welcome to In-Sight™, a compact, easy-to-use, stand-alone intelligent vision sensor for industrial machine vision, process control, and factory automation applications. In-Sight can perform inspection, identification, measurement, and other similar tasks. It is, in many respects, a dedicated, freestanding vision computer. About the size of a paperback book, it performs all of its own processing. It can acquire images, display them, control lights, and communicate with external devices. Its innovative spreadsheet interface and built-in vision software greatly simplify the task of developing and maintaining vision applications.

For detailed information, open one of these topics:

- Introduction
- Using the Control Pad
- Modes
- Working with Images
- Worksheet and Vision
- Using the In-Sight Server
- Clocked Data
- Geometry
- Image Processing
- PatFind™
- Text Functions
- Release Note

- Installing In-Sight
- Using the In-Sight Interface
- Working with Jobs
- Using the Worksheet
- Setting Options
- Image Acquisition
- Coordinate Transforms
- Graphics
- Input/Output
- Structures
- Vision Data Access
- Glossary

- Getting Started
- Online and Offline
- Saving Settings
- Building Formulas
- Building an Operator Interface
- Blob Analysis
- Edge Detection
- Histogram Analysis
- Mathematics
- System Functions
- Hardware Reference

NOTE This is a rough draft. Its content is incomplete, inaccurate, and unreviewed. It should give you the flavor of the forthcoming document, but you can rely on it only approximately.

NOTE This confidential document is for Cognex internal use only. The information provided here is subject to change without notice. Do not circulate without proper authorization from the program manager. Please report any bugs, problems, or concerns about the hardware, software, or documentation to Tom Baker at the Portland office by internal email.

COGNEX In-Sight

◄ ▲ ► Intro: Overview

These topics offer introductory information about In-Sight:

- What is the In-Sight Vision Sensor?
- Features and Capabilities

- Vision Functions
- Developing and Deploying Applications
- Core Vision Processing Steps
- Customer Support.

COGNEX In-Sight

◄ ▲ ► Intro: What is In-Sight?

The Cognex In-Sight vision sensor is a low-cost, easy-to-use, stand-alone intelligent camera designed for industrial machine vision, process control, and factory automation applications. It can perform inspection, identification, measurement, and other similar tasks. It is, in many respects, a compact, dedicated, freestanding vision computer. About the size of a paperback book, it performs all of its own processing. It can acquire images, display them, control lights, and communicate with external devices:
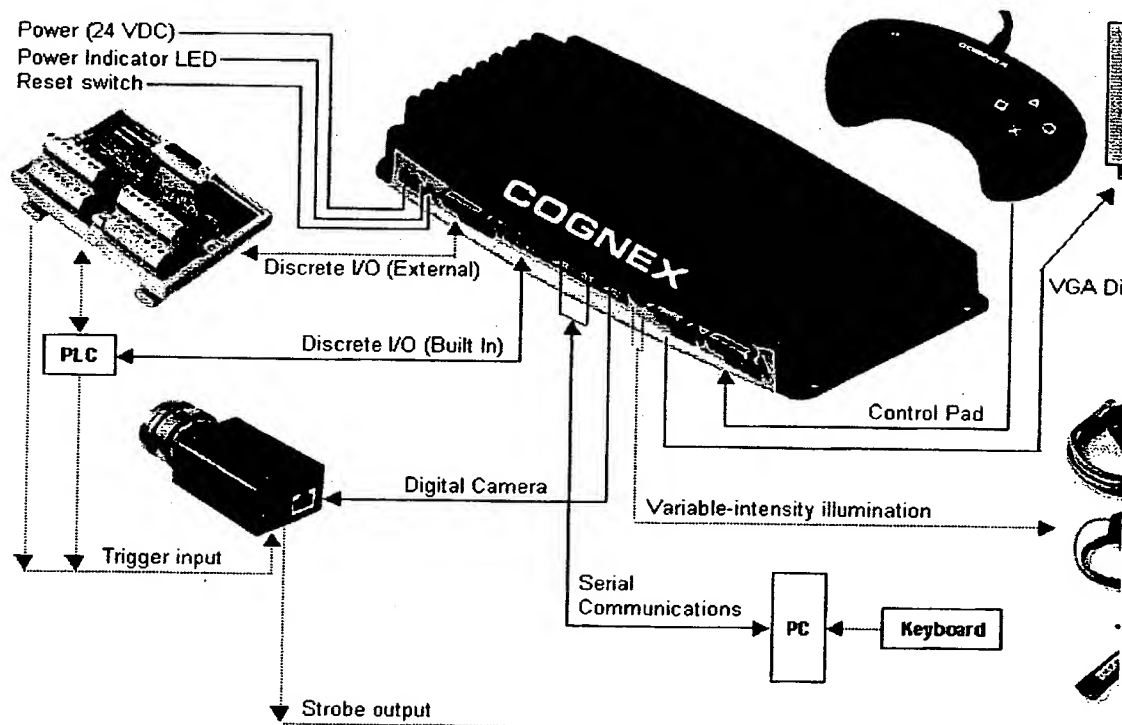
Unlike vision systems based on frame grabbers, In-Sight can operate independently, without a host PC, monitor, or keyboard. Unlike traditional sensors—photocells and the like—it offers sophisticated vision processing algorithms and significant computational power. In-Sight bridges the gap between sensors and

computers and offers the reliability required on the real-world factory-floor.

In-Sight is easy to configure and reconfigure. Instead of conventional programming, it has a spreadsheet with built-in image processing functions. Instead of a keyboard, it has a Control Pad, similar to the game pads used with video games. When setting up an application, you attach a monitor and a Control Pad, viewing an image to set up image acquisition, image processing, feature extraction, and so on. In a finished application, you can remove the monitor, Control Pad, or both and run the In-Sight sensor as freestanding device. In-Sight then receives commands and sends results via its I/O and RS-232 interfaces.

Setting up In-Sight requires a knowledge of the relevant manufacturing process but does not require a deep immersion in vision processing. Typical In-Sight tasks include gauging dimensions, locating parts, inspecting assemblies, and so on.

COGNEX In-Sight

◄ ▲ ► Intro: Features and Capabilities

- Standalone operation. No PC required for setup or run-time.
- Straightforward graphical interface run through an eight-button Control Pad. Reflecting the factory floor environment, this pendant controller lets an operator run the system even while standing, wearing gloves, away from a desk, or away from a monitor.
- Spreadsheet-based vision processing interface—drawn directly on the image, with adjustable transparency—for easy configuration. In-Sight presents complex image data and results through worksheet cells as simple scalar values that can be manipulated with conventional spreadsheet functions.
- Built-in functions for constructing worksheet formulas:
  - o Feature-extraction and image analysis techniques including blob analysis, edge detection, pattern matching, geometric measurement, histogram analysis, coordinate transforms (calibration), and so on.
  - o Image-processing operations including neighborhood filters (high pass, low pass, erode, dilate, open, close, tophat, and bothat) and point filters (binarize, clip, equalize, and stretch).
  - o Image comparison, which checks an ROI against a template and creates a result image that highlights the differences between the ROI and the template.
  - o Mathematical functions and operators, providing a full range of standard arithmetical, conditional, logical, statistical, and trigonometric operations.
  - o Text functions, to manipulate text strings for display or serial communications.
- Clocked data functions, used to increment a value or to delay an output signal for a specified time, typically to wait for an inspected item to reach a piece of equipment triggered by the output signal.
- Graphics drawn on the display. Available graphics include:
  - o Charts, used to monitor the history of a value.
  - o Controls drawn on the worksheet such as buttons, list boxes, and check boxes, commonly used to build an operator interface.
  - o Geometric shapes drawn on the image.
- Fast, compact hardware, purpose-built for image processing. Includes 4 MB of flash memory (for

nonvolatile job and program storage) and 16 MB SDRAM (for run-time processing and image acquisition).

- Industrial packaging: rugged cast-aluminum enclosure; no moving parts such as fans or disk drives; ? flanged? attachment points for horizontal or vertical mounting.
- Digital camera producing a high-quality 8-bit 640x480 image. Offers rapid reset, progressive scan, full-frame integration, and a C-mount lens.
- Dynamic light control for one or two external variable-intensity lights, each with one or two light banks.
- Versatile I/O:
    o Two RS-232C serial ports.
    o Ten discrete inputs and ten discrete outputs, for general-purpose use.
    o On-camera trigger input and strobe output.
- Server program that runs on a Windows PC, to save images and archive jobs over the serial port. Includes a terminal program with commands to acquire images, run jobs, get and set cell values, and so on.
- Standard 15-pin DIN connector for a VGA monitor or a compatible flat-panel LCD display.

═══ **COGNEX In Sight**

◄ ▲ ► Intro: Vision Functions

The In-Sight software includes built-in vision processing functions, mathematical functions, and operators, all used to build formulas in the In-Sight worksheet. The categories are:

Category	Description
Acquisition	The AcquireImage function acquires an image and stores it in an Image structure in cell A0. It also sets values that control the lights and camera including trigger source (manual, continuos, on-camera, discrete I/O, or serial, and so on), the first scan line to acquire, the number of scan lines to acquire, and video gain and offset.
Blob Analysis	Finds, analyzes, and classifies *blobs*, which are contiguous patches of pixels on one side of a grayscale threshold. Blob analysis—also called connectivity analysis—provides data about the number, location, shape, and orientation of blobs. Vision applications use blob analysis for feature extraction, especially for features defined by generalized criteria such as an area or perimeter range.

In-Sight's blob functions, listed under Vision Processing in the Formula Builder, offer automatic thresholding and the ability to select blobs by color, size, area, and other characteristics. |
| Clocked Data | Offers time-related functions that:

- Keep various kinds of running totals.
- Store a value over a time interval, for example to delay an output signal to synchronize it with a device deeper in the production line. |

Coordinate Transforms   Maps between coordinate systems. The Formula Builder lists three kinds of transformations:

- **Calibrate.** Establishes a coordinate transformation between two planes. Often used to create a <u>calibrated</u> transformation, which converts in either direction between image locations (in Pixel coordinates) and physical locations (in World coordinates). For example, an application might convert points and distances from row-and-column pixel values to real-world units such as millimeters or inches. To simplify the process of converting feature extraction and measurement results, In-Sight offers functions that convert entire structures in one step.

  Although through-the-lens calibration is a common use for coordinate transformation, you can use In-Sight's calibration functions for other plane-to-plane transforms. Because In-Sight computes the transform from four known points, the two coordinate systems can be skewed.

- **Fixture.** Converts between a reference (Fixture) coordinate system and either World or Pixel values. A *fixture* is a reference coordinate system (X, Y, theta) located in an image and used to define other locations through offsets, typically to compensate for image-to-image variation in position, orientation, or both.

- **Warp.** Maps a point in a warped image back to its location in the original, unwarped image. (WarpImage, under Vision Processing, constructs the warped image.)

Edge Detection   Detects straight or curved edges, each marked by a light-to-dark or dark-to-light transition with a specified width and strength. Vision applications use edge detection to extract and measure linear features.

In-Sight's edge-detection functions, listed under Vision Processing, offers methods for filtering false edges, excluding uninteresting edges, and sorting them in various orders. For straight edges or edge pairs, In-Sight measures the distance between the edges. For circular or curved line edges, In-Sight measures the radius.

Geometry   Measures distances and angles and constructs geometric shapes. Categories are:

- **Measure:** gauges distances and angles of points, lines, and circles in various combinations. These basic geometric measurements are crucial for many inspection applications, which use them to check the dimensions and locations of parts.

- **Fit:** constructs geometric shapes.

Graphics

- Draws graphics on the image including arcs, circles, lines, and regions.
- Adds graphic controls to the worksheet such as buttons, check boxes, and list boxes. You typically use these controls to permit operator input in the deployed application. They remain active even when you lock the worksheet.
- Displays charts (which plot the history of a value, to determine correct parameter settings, or for process monitoring) and a status indicator (color coded for Pass/Fail/Warning). Both are commonly used to operator output in the deployed application.

Histogram Analysis	Counts grayscale values from an image region and plots their grayscale distribution. Values emitted to the worksheet include an automatically computed binary threshold (the grayscale value that best separates the image into light and dark regions) and averages for the light region, dark region, and entire histogram. Statistics available for all or part of the histogram include the number of pixels, the first and last non-zero gray level (head and tail), the most and least common gray levels (maximum and minimum), the sum, and the standard deviation. Vision applications commonly use histograms for image analysis or feature extraction. Listed under Vision Processing.
Image Processing	Collects operations that yield an image result:

- **Image Processing**, which offers neigborhood filters (high-pass, low-pass, erode, dilate, open, close, tophat, bothat, edge direction, and edge magnitude) and point filters (binarize, clip, equalize, stretch).
- **Image comparison**, which compares an image with a template to detect their differences. A common use is detecting anomalies in inspection applications.
- **Warping**, a polar transformation that unwraps a curved or circular feature into a second, rectangular image. (Coordinate Transforms has a function that maps a point in the warped image back to the corresponding location in the original image.)

Input/Output	Reads and writes to external devices through discrete I/O or the serial ports.
Mathematics	Lists mathematical functions and operators in the following groups:

- **Logic**: logical AND, OR, and NOT; also range checking.
- **Lookup**: finds values in lists. Includes If(), for conditional formulas.
- **Math**: arithmetical functions—absolute value, exponent, random number, square root, sum, and so on.
- **Operators**: Mathematical, logical, and relational operators. Their precedence is the same as ANSI C.
- **Trigonometry**: Cos, Sin, Tan, Pi, Radians, and so on.
- **Statistics**: Maximum, Mean, Minimum, and Standard Deviation.

PatFind™	Performs pattern-matching searches after building model. In-Sight supports two kinds of model, edge-based and area-based:

- An edge model extracts geometric boundaries from an image region and uses them to find instances of the model in an image. Edge models tolerate variations in rotation, scale, and image intensity; they also tolerate partial occlusions.
- An area model extracts specified image area to use as a template. It then measures the similarity between the template and a series of image regions through normalized correlation, evaluating grayscale differences on a pixel-by-pixel basis. An area model tolerates variation in rotation and scale.

Vision applications use pattern matching for feature extraction, particularly for features that are stable in size and shape. The PatFind functions are listed under Vision Processing.

Structures	Creates storage for fixtured circles, cross marks, points, or regions. All of the values that define the structured item reside in a single cell, instead of consuming one cell each, simplifying the worksheet. Some applications define structures for "global" items, since changing a value in a structure changes it in all formulas that refer to it.
Text Functions	Manipulates text strings. For example, finding text within a string, replacing text in a string with new text, converting a string to lowercase or UPPERCASE, comparing two strings, and so on. Common uses are manipulating strings for serial communications or for on-screen display.
Vision Data Access	Gets individual values from the structures generated by vision functions. Each category— Blobs, Edges, Hist, and so on—lists the subset of result functions available for that structure.

COGNEX In-Sight

◄ ▲ ► Intro: Develop and Deploy Applications

The In-Sight vision sensor has many applications in inspection, quality control, and factory automation. A typical application starts off in an office or lab environment and then moves to the factory floor for integration and testing. After connecting and adjusting all of the parts of the system—including the vision processing equipment, the mechanical subsystems, and so on—the In-Sight hardware runs continuously in the production environment. If desired, you can lock In-Sight's configuration to prevent unauthorized change. The following figure shows a production line with an In-Sight system:
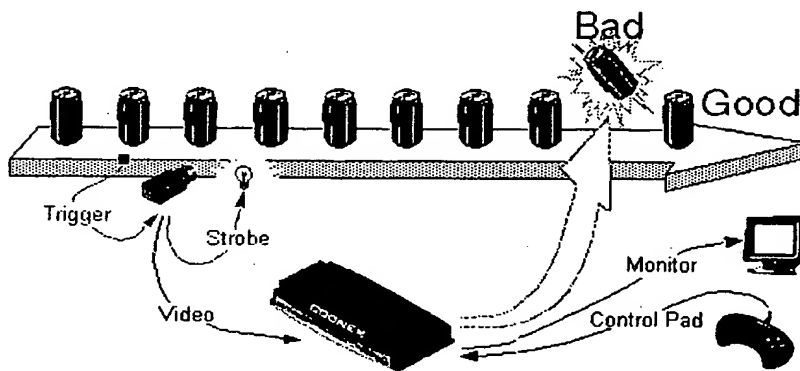
The exact development steps depend on the task at hand. A typical sequence might be:

1. Install the In-Sight system on a test stand. Add other equipment such as a monitor, a trigger source, and so on.
2. Determine the requirements of the vision application. Preliminary steps might include reviewing quality control and inspection specifications, interviewing production personnel, evaluating I/O requirements, evaluating operator interface requirements, and generally understanding the manufacturing context. For an inspection application, for example, it is important to fully characterize "good" parts, "bad" parts, and the differences between them.
3. Select a C-mount lens. The focal length usually depends on the desired field-of-view and working

distance:

- o *Field-of-view* is the area seen through the lens. In general, it must be large enough to resolve the features of interest including an adequate positional tolerance—but small enough to avoid wasting resolution on unimportant objects.
- o *Working distance* is the separation between the camera and the object. Physical or mechanical constraints often influence the working distance. For the same field of view, increasing the working distance requires increasing the focal length of the lens, reducing depth of field. Sometimes you can use mirrors or prisms to improve camera placement.

For more information, see *Selecting a Lens*.

4. Select lighting. Lighting technique plays a very large role in an application's success. Considerations include:

- o The light level must be bright enough to illuminate the features of interest at the working aperture and shutter speed. You can sometimes compensate for insufficient light by increasing aperture or gain, but providing adequate light is almost always the first-choice solution for solving exposure problems.
- o The type of light must be appropriate for the features of interest and the purpose of the application. Light can be white or colored, diffuse or directional, on-axis or off-axis, oblique or perpendicular, unidirectional or omnidirectional, continuous or strobed, and so on. The wide range of choice makes it difficult to offer generalized suggestions, but it is vitally important to select lights that makes the important details prominent in the image.

For more information, see *Lighting Guidelines* and *Lighting Techniques*.

5. Collect representative "good" and "bad" part samples, and acquire a set of test images. The acquisition settings determine exposure; one strategy is to adjust the lights and image sensor to place the features of interest near the center of the grayscale range, seeking the highest-contrast image that does not saturate the brightest or darkest values.

6. Set up a worksheet that performs the necessary vision processing. Image-related operations include:

- o Image processing operations such as filters, image warping, and image arithmetic.
- o Feature extraction operations such as blob analysis, edge detection, and pattern matching.
- o Feature analysis operations such as histograms, distance measurement, and angle measurement.

A common goal at this stage is to obtain reliable results in a lab environment. For a representative sequence of processing steps, see *Core Vision Processing Steps*.

7. Most In-Sight applications communicate with other production equipment through serial I/O, discrete I/O, or both. Depending on the source of the acquisition trigger, data signals from other equipment, and the method for communicating results, you configure the worksheet for the necessary processing:

- o For serial communications with a PC or other serial devices—typically to receive commands, return results, or archive jobs—connect the device to one of the Vision Processor's serial ports and configure In-Sight for its baud rate, parity, and so on. To manipulate strings read or written to a serial port, use In-Sight's text functions.
- o If your application uses discrete I/O—typically to signal PLDs and other production equipment—then connect I/O through twisted pair or other suitable wiring. Next, configure the worksheet to read or write the associated bits and respond appropriately when they change state. To delay an output signal, typically to wait for an inspected item to reach a piece of equipment triggered by the signal, use In-Sight's DelayLine() function.

It is not always practical to develop the I/O-related processing in a lab environment, away from the actual production equipment.

8. If desired, construct an <u>operator interface</u>. In-Sight lets you put buttons or other graphical controls in cells for end-user input or output. It also lets you simplify, customize, or hide the worksheet. When designing the interface, it can be helpful to seek feedback from the technicians who will run the system.

9. <u>Save the job</u> to flash memory.

10. Move the In-Sight system to the factory floor, connecting it to the lights, trigger source, power source, Control Pad, monitor, sensors, PLDs, and so on.

11. Test the system in the production environment, optimizing it for accuracy and speed. Goals include:
    o Validating the system's behavior over a wide range of normal and abnormal conditions.
    o Confirming that the worst-case execution time is less the longest allowable cycle.
    o Examining false accepts or rejects. Recording images associated with inaccurate results can help you diagnose and correct problems.

12. If desired, <u>lock the finished job</u>, preventing unauthorized changes.

13. Save the finished job, preserving the final parameter values.

14. <u>Archive</u> the contents of flash memory to a PC. You can use the archive as a safety copy or for In-Sight–to–In-Sight job transfer.

15. Run the production system.

COGNEX In-Sight

◄ ▲ ► Intro: Core Vision Processing Steps

Vision applications differ in purpose and strategy but often have roughly the same underlying structure. This topic lists a representative sequence of steps for the vision processing part of an application.

1. <u>Acquire an Image</u>. A typical application snaps a new image in response to a trigger signal generated by a device that detects the presence of the item of interest. For other inputs, In-Sight also has ten discrete inputs—two in and two out on the Vision Processor, and eight in and eight out on the outboard I/O Expansion Module.

2. Configure the System. Many applications require one-time or occasional operations such as <u>building a model</u>, establishing a <u>calibrated transformation</u>, and so on. In-Sight lets you conditionally or unconditionally <u>disable cells</u>, freezing their contents. Configuration steps are good candidates for conditional updating.

3. <u>Fixture the Part</u>. Manufacturing operations often do not perfectly constrain the location and orientation of the parts being inspected. Machine vision applications therefore must tolerate positional and angular uncertainty. A common strategy is to establish a <u>fixture coordinate system</u> in each image. Many applications identify the fixture origin and angle through pattern matching, edge detection, or another feature extraction technique.

    After finding the fixture origin—which might be a mark drawn on a part for this purpose, a distinctive

Appendix I - Page **9**

feature in the image, or a computed point—the application saves its row, column, and angle values. Subsequent operations refer to the fixture values to compensate for image-to-image differences in location and orientation. Some applications find a new fixture coordinate system for each image; others update in only on failure.

4. Define a Region of Interest (ROI). An ROI is an image area for image processing or feature extraction. An ROI is usually rectangular but sometimes angled, curved, circular, or toroidal. Its center is commonly the expected feature center. Its size depends on the feature size, positional uncertainty, angular uncertainty, and other factors. Keeping the ROI as small as practical reduces the number of pixels to process, reducing processing time.

   In-Sight supports several strategies for defining ROIs. An ROI can be:
   o Fixed. Its parameters are unchanging values.
   o Interactive. You can set its parameters with a graphic cursor drawn on the image and manipulated with the Control Pad.
   o Calculated. The ROI values derive from worksheet formulas that adjust the ROI at run time.
   o Fixtured. The ROI is defined by offsets from a fixture coordinate system. A special case of "calculated."

5. Process the ROI Pixels. In-Sight offers a full compliment of built-in image processing and image arithmetic algorithms. Finding the right combination of preprocessing steps can be a complex aspect of vision development. A high-pass filter, for example, strips out low-frequency information—which might be "signal" in one context but "noise" in another. Appropriate image enhancement can greatly improve the application's reliability.

6. Detect features. After enhancing the image, a typical application scans the ROI for the feature(s) of interest. In-Sight offers several feature extraction methods including edge detection, blob analysis, and histogram analysis, pattern matching. The choice of method depends on the characteristics of the feature in question. A linear feature, for example, might suggest the use of edge-detection.

   For example, if you are inspecting a part that should have three holes of the same size arranged in an equilateral triangle, then you expect the image to have three dark circles corresponding to the holes. Your application might use edge detection to look for circular edges with the expected radius of curvature and light-to-dark transitions.

7. Analyze features. After detecting features, the application usually evaluates them to derive useful information. A common goal is to validate the feature data against some expected profile. This feature analysis occurs in the worksheet, which offers mathematical and logical functions and operators, measurement functions, the ability to chart values over time, and other resources for data analysis.

   Continuing the example, after enhancing the image, the application could test the individual dots to confirm that they are the right size. Next, it could count them to confirm that exactly three are present, failing the inspection if there are missing or extra dots. It could then measure all combinations of angles and distances to confirm the equilateral arrangement.

8. Draw the results on the image. If the application includes a monitor that displays the image, it might draw text or symbols on the image.

9. Respond to the results. After examining the image features, most applications make a decision of some kind and take action if necessary. For inspection, the result is generally a pass-fail signal. For process control, it might be a numerical score passed to another device for evaluation and possible action.

   Continuing the example, if the feature analysis shows that the part has three holes in the right places,

then it passes the inspection and continues down the line. But if it fails a test, then it is a reject and removed from the line. In-Sight has ten discrete outputs; you might connect one to a device that removes defective parts from the production line. If necessary, you can delay the output in a buffer. If necessary, you can use calibration to map image locations to their real-world equivalents.

10. Back to step one.

Many real-world applications are elaborations on this flow of events. To consider these steps in a broader context, see _Developing and Deploying Applications_.

COGNEX In-Sight

◄ ▲ ► Intro: Customer Support

For options and accessories, call Cognex at 877-855-8705. For technical problems or questions, contact Cognex Technical Support through one of the methods listed below.

Location	Contact
USA	**Cognex Corporation, Corporate Headquarters**
	One Vision Drive / Natick, MA 01760
Main	Tel: 508-650-6300; Fax: 508-650-3321
	Email: tech_support@cognex.com
	Hours: 8:30 am to 8:00 pm Eastern Standard Time, Monday through Friday
Midatlantic	**Cognex Corporation, Midatlantic Regional Technology Center**
	530 East Swedesford, Suite 404 / Wayne, PA 19087
	Tel: 610-975-9592; Fax: 610-975-9616
Midwest	**Cognex Corporation, Midwest Regional Technology Center**
	850 East Diehl Road, Suite 160 / Naperville, IL 60563
	Tel: 630-505-9990; Fax: 30-505-9995
Southwest	**Cognex Corporation, Southwest Regional Technology Center**
	100 Congress Avenue, Suite 2100 / Austin, TX 78701
	Tel: 512-370-5245; Fax: 512-370-5247
West Coast	**Cognex Corporation, West Coast Regional Technology Center**
	1001 Rengstorff Avenue / Mountain View, CA 94043
	Tel: 650-969-4812; Fax: 650-969-4818
France	**Cognex France**
	Immeuble le Patio / 104 avenue Albert 1er / 92563 Rueil Malmaison / France
	Tel: 011-33–1-47-77–15-50; Fax: 011-33–1-47-77–15-55
	Email: europe_support@cognex.com
Germany	**Cognex Germany**
	Greschbachstrasse 12 / 76229 Karlsruhe / Germany
	Tel: 011-49-721-96187-0; Fax: 011-49-721-61566–1

Italy	**Cognex Italy**
	Via Gasparotto, 1
	I-20124 Milano - MI Italy
	Tel: +39-02-67471200; Fax: +39-02-67471300
Japan	**Cognex CKK**
	2-28-8 Honkomagome, Bunkyo-ku
	Tokyo 113-6591, Japan
	Tel: 011-81-3-5977-5400; Fax: 011-81-3-5977-5401
	Email: ckk_support@cognex.com
Korea	**Cognex Korea, Inc.**
	#805, DongKyung Building, 824-19
	YukSam-Dong, KangNam-Ku
	Seoul 135-080, Korea
	Phone: 011-82-2-539-9047 or 011-82-2-539-9048; Fax: +82-2-569-9823
Singapore	**Cognex Singapore**
	Block 51 Ayer Rajah Crescent
	#06-08/09 Ayer Rajah Industrial Estate
	Singapore 139948
	Tel: 011-65-773-1261; Fax 011-65-773-1423
UK	**Cognex UK**
	Chancery House / 199 Silbury Boulevard / Milton Keynex, MK9 1JL
	United Kingdom
	Tel: 011-44-1908-206000; Fax: 011-44-1908-392463

COGNEX In-Sight

◄ ▲ ► Install: Overview

This section describes how to install the In-Sight hardware and software. Please read the following topics before attempting the installation:

- Standard Components
- Optional Components
- Preparing for Installation
- Installing the Hardware
- Installing the Software

- Testing the Installation
- Testing the Lights
- Updating the Firmware
- Starting Up and Shutting Down
- Troubleshooting

The general installation sequence involves connecting the Vision Processor with the Control Pad, camera, monitor, and power supply as well as others devices such as variable-intensity lights, an external trigger, strobe lights, and so on. When done, you should be ready to start developing your In-Sight applications.

CAUTION
- Incorrect voltage creates a risk of fire or shock and can damage the In-Sight hardware. Never connect a power source other than 24v DC to the Vision Processor. When connecting the power source, carefully observe the correct polarity.
- Avoid locations that expose the In-Sight hardware to excessive heat, dust, moisture, humidity, impact, vibration, corrosive substances, flammable substances, static electricity, or other environmental hazards. Any of these hazards can damage the In-Sight hardware.
- ??To reduce the risk of damage or malfunction due to overvoltage, noise, power surges, electrostatic discharge (ESD), or other irregularities:
    - o Route all cables as far away as practical from high-voltage power sources.
    - o Connect an earth ground to the Vision Processor chassis. If you also ground the camera, then the Vision Processor chassis and camera chassis must have exactly same ground potential.
    - o When handling electronic components, observe ESD precautions, such as the use of an anti-static wrist strap connected to a flat, grounded work surface.
- Before connecting or disconnecting any equipment to the In-Sight hardware, remove the power from In-Sight and from the equipment being connected or disconnected.
- In-Sight has a dedicated digital camera. Other cameras are incompatible. Connecting an off-the-shelf camera to the Vision Processor can damage the camera, Vision Processor, or both.
- Do not open the In-Sight Vision Processor, camera, or other hardware, which contain no user-serviceable internal parts. Do not make unauthorized electrical or mechanical modifications to the In-Sight hardware.

COGNEX In-Sight

◄ ▲ ► Install: Standard Components

The hardware components supplied with In-Sight depend on your configuration and options. Before you install In-Sight, compare the contents of the shipping container with those listed in packing slip. If anything is missing or damaged, call the shipping company. Also call Cognex as described in *Customer Support*.

Component	Description	Part Number
Vision Processor	System unit in cast aluminum housing.	800-5714-2
Control Pad	Pendant controller with four buttons plus a two-axis cursor. Integral 10-foot cable terminated with a DB-15 connector.	800-5713-1
Camera	Dedicated digital camera.	118-0034
Camera cable	25-foot Category 5 shielded twisted pair (STP) cable terminated with standard, 8-pin RJ-45 connectors.	XXX-YYY
Camera cable strain reliever	Flexible hood that protects the cables.	XXX-YYY
Ferrite core	Attached to Camera Cable, for European CE compliance.	XXX-YYY
Trigger Terminal Block	4-pin terminal block for connecting trigger and strobe to the camera.	Cognex: XXX-YYY Phoenix: #1803594
Discrete I/O Terminal Block	8-pin terminal block for connecting discrete I/O to the Vision Processor.	Cognex: XXX-YYY Phoenix: 1803633
Power Terminal Block	2-pin terminal block for connecting power to the Vision Processor	Cognex: XXX-YYY Phoenix: 1803578
CD-ROM	Software and documentation.	XXX-YYY
Getting Started With In-Sight	Hardcopy user guide.	XXX-YYY
Installing In-Sight	Hardcopy installation guide.	XXX-YYY

COGNEX In Sight

◄ ▲ ► Install: Optional Components

Component	Description	Part Number
C-mount Lenses	6.0 mm	114-0049
	8.5 mm	114-0039
	12.5 mm	114-0004
	16.0 mm	114-0018
	25.0 mm	114-0001
	35.0 mm	114-0041
	50.0 mm	114-0040
Extension tubes	Set containing 0.5, 1.0, 5.0, 10.0, 20.0, and 40 mm extension tubes.	114-0009

Variable Lights	Back light	XXX-YYYY
	Dark Field light	XXX-YYYY
	Dome light	XXX-YYYY
	Linear Array light	XXX-YYYY
	On-Axis Diffuse light	XXX-YYYY
	Ring light	XXX-YYYY
	Spot light	XXX-YYYY
Light Cable	Y-adapter cable, 9-pin DIN to 7-pin DIN.	XXX-YYYY
Light Cable	Y-adapter cable, 9-pin DIN to RJ-11	XXX-YYYY
Strobe	Interface box required for strobed operation.	XXX-YYY
Control Module		
Strobe cable	Supplies power to Cognex strobes.	XXX-YYYY
Serial cable	??10-foot shielded Category 5 cable, terminated with standard, 8-pin RJ-45 connectors.	XXX-YYY
Serial adapter	Converts the RJ-45 output to DB-9 format.	XXX-YYY
AC adapter	Converts 100v–240v AC line current (50–60 Hz) to 24v DC current. Three-foot leads on the DC side; removable AC power cord. ??different cords for different countries??.	XXX-YYY
I/O Expansion module	Connects up to 8 additional discrete inputs and outputs.	800-5712-1
I/O Module cable	7-foot cable to connect the I/O module to the Vision Processor.	300-0274-7

COGNEX In-Sight

◄ ▲ ► Install: Preparing for Installation

To complete the installation, you must be familiar with your hardware and software environment. Items that you might need include:

- A 2 mm flat-head screwdriver, to tighten the screw on the terminal blocks.
- Four ¼-inch bolts or other fasteners, to attach the Vision Processor to a convenient surface.
- ?? ??-inch bolts or other fasteners, to attach the camera to a convenient surface.
- A VGA-compatible monitor, to view images from In-Sight
- A PC running Windows NT ?4.0? or higher, to run the In-Sight server software, used to archive jobs, archive images, issue commands by keyboard, and so on.
- A PC running Internet Explorer 4.0 or higher, to view the online documentation, distributed as a compiled HtmlHelp (.chm) file.

For information about obtaining lenses, lights, and other In-Sight options, contact Cognex.

COGNEX In-Sight

◄ ▲ ► Install: Hardware

This section details how to install each piece of hardware, discussing the following topics:

- Attaching the Vision Processor
- Connecting the Control Pad
- Connecting the Camera
- Connecting an External Trigger
- Connecting a Strobe Light
- Connecting Variable-Intensity Lights

- Connecting a VGA Monitor
- Connecting a Serial Device
- Connecting Built-in Discrete I/O.
- Connecting External Discrete I/O.
- Connecting a Power Supply
- Troubleshooting
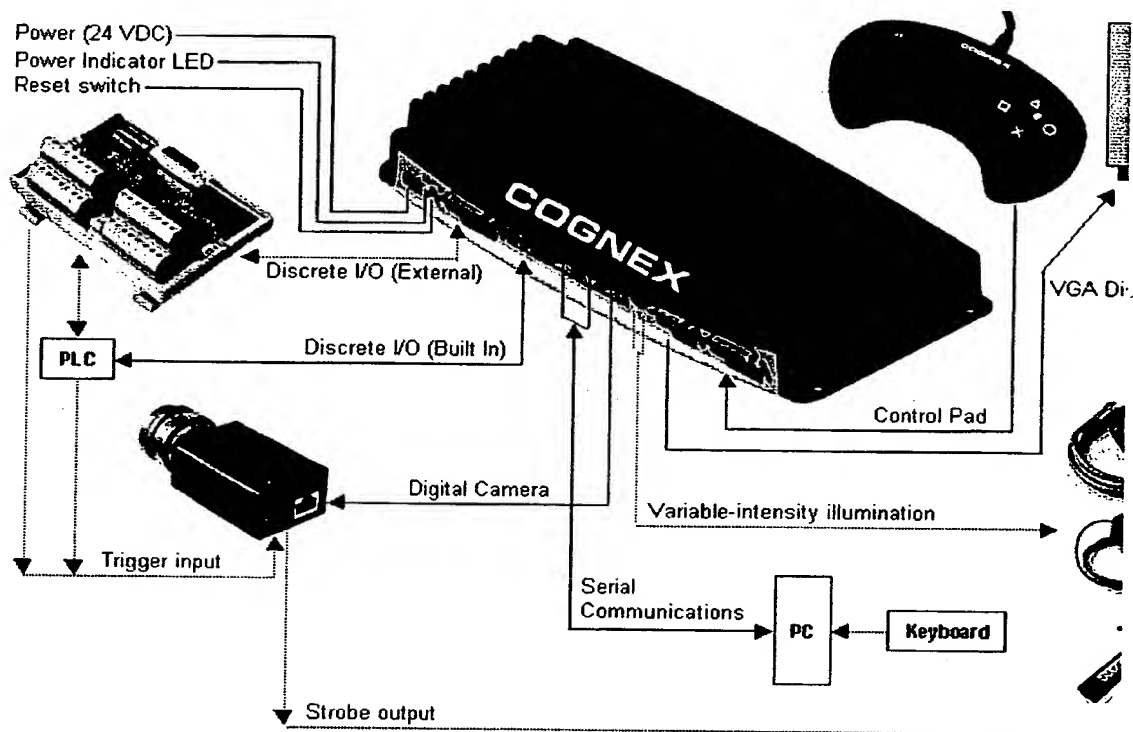
◄ ▲ ► Install: Vision Processor

You can bolt the Vision Processor (P/N 800-5714-2) to almost any convenient surface using ¼-inch or

smaller fasteners. General considerations in choosing a location include satisfying its environmental requirements and practical constraints such as cable routing. Due to its small size and generous cable lengths, you enjoy great flexibility when locating and orienting the Vision Processor.

Horizontal orientation   Vertical orientation

Four 1/4-inch mounting holes

CAUTION Avoid locations that expose the In-Sight hardware to excessive heat, dust, moisture, humidity, impact, vibration, corrosive substances, flammable substances, static electricity, or other environmental hazards. Any of these hazards can damage the In-Sight hardware.

▶ **To Connect the Vision Processor**

1. Choose a location:
   o Try to mount the Vision Processor in the open or in a well-ventilated cabinet. Do not impede its air circulation or block its cooling fins. Mounting the Vision Processor "upside down" reduces its cooling efficiency.
   o Prefer a horizontal or vertical orientation that provides easy access to the front panel and that minimizes tangling among the wires. One possibility is to attach it to a table with the connector edge in front; another is to attach it to a wall with the connector edge parallel to the floor.
   o The Vision Processor's base is uncoated stainless steel, so the electrical and thermal properties of the adjacent surface can affect the Vision Processor. Bolting it directly to a conductive, grounded surface is one way to provide an electrical ground. A cool surface might provide a supplemental heat sink.
2. Remove the 24v DC power from the In-Sight system. To reduce the risk of malfunction or damage, never connect or disconnect In-Sight components when power is present.
3. Pass ¼-inch (or smaller) bolts through its mounting holes and into a solid surface.
4. Connect an earth ground to the Vision Processor chassis. If you also ground the camera, then the Vision Processor and camera must have exactly same ground potential.
   NOTE The Vision Processor chassis is not intrinsically grounded. Its potential floats unless you supply a known ground.

For more information about the Vision Processor, see the *In-Sight Hardware Reference*.

COGNEX In-Sight

◀ ▲ ▶ Install: Control Pad

When you develop In-Sight applications, you connect a Control Pad (P/N 800-5713-1), which has an integral 10-foot cable. A deployed application that does not require operator input might retain or omit the Control Pad.

Control Pad

CONTROL PAD port
(DB-15 connector)
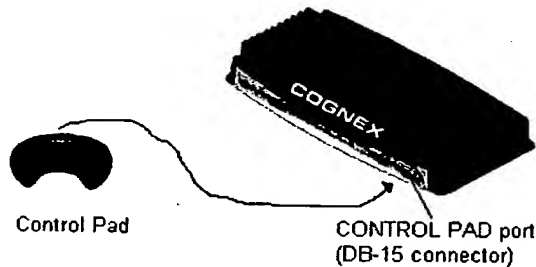
▶ **To Connect the Control Pad**

1. Remove power from the Vision Processor.
2. Plug the male DB-15 connector at the end of the Control Pad cable into the Vision Processor's CONTROL PAD port.

In-Sight automatically calibrates the attached Control Pad on power up. To calibrate it manually, simultaneously press the ✕, ▢, △, and ○ buttons.

Most analog, 2-axis, 4-button PC game pads should work with In-Sight, which uses a standard DB-15 game port pinout. If you lose or damage the In-Sight Control Pad, you can often reduce down time by substituting an off-the-shelf game pad. Cognex does not test In-Sight with non-Cognex game pads, however, and cannot assure their compatibility. Special features such as "turbo mode" are incompatible with In-Sight. You can order spare or replacement Control Pads from Cognex at 877-855-8705.

COGNEX In-Sight

◄ ▲ ► Install: Lens and Camera

The In-Sight camera sends a video signal to the Vision Processor over a Category 5 shielded twisted pair (STP) cable terminated with standard RJ-45 connectors. The maximum cable length is 25 feet. Installation involves mounting the camera and connecting the cable to the camera and Vision Processor. This section describes how to install the camera and optics.

C-mount lens

Mounting block

4-pin Terminal Block

Strobe output

Camera Cable

Ferrite core

Trigger input

CAMERA port
(RJ-45 connector)
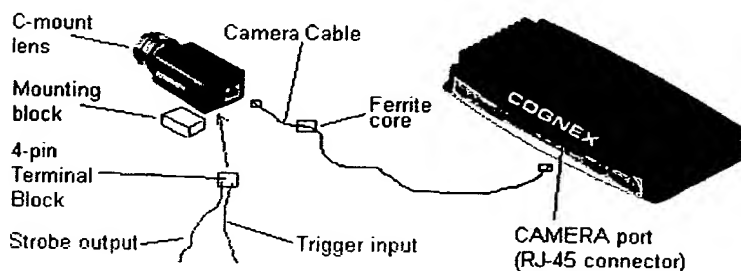
CAUTION   • Avoid locations with excessive moisture, humidity, heat, dust, impact, and other hazards, which can damage the camera.
          • The camera chassis must have the same ground potential as the Vision Processor chassis. Any difference in potential creates a ground loop that can damage the camera, Vision Processor, or both. Using the non-conductive camera mounting block electrically isolates the camera from the mounting point, allowing the Vision Processor to supply a ground to the camera through the camera cable. In general, avoid attaching the camera directly to a conductive surface to avoid the risk of a ground loop.
          • The Vision Processor has three RJ-45 connectors, one for the camera and two for the serial ports. Make sure you plug each device into the correct connector. Plugging the camera into a serial port might damage the camera, the Vision Processor, or both.

▶ **To Mount the Camera**

1. Choose a camera location based on the features of interest, optics, desired working distance, and so on. The mounting point should be as solid and stable as practical to avoid vibration, which can blur images.
2. Screw a C-mount or CS-mount lens into the camera. Focal length depends on working distance, field-of-view, feature resolution, and other considerations. See _Selecting a Lens_ and the _Field-of-View Table_.
    NOTE   If using a CS-mount lens, add a 5 mm extension tube to compensate for the optical differences between C-mount and CS-mount systems. When calculating field-of-view, be sure to allow for the extension tube.
3. If the camera is not attached to the non-conductive mounting block, attach the mounting block to the camera. (When shipped, the mounting block is attached, so this step is usually unnecessary.)
4. Attach the mounting block and camera to a mounting post or other solid attachment point by means of the ¼-20 threaded attachment point on the bottom of the mounting block.
5. Remove power from the Vision Processor.
6. ??To conform to European CE regulations, attach the ferrite core to the camera cable.
7. Plug one end of the camera cable into the camera's RJ-45 connector and the other end into the Vision Processor's CAMERA port.
8. Connect wiring for a trigger input, strobe output, or both through the detachable 4-pin terminal block.
9. To prevent strain on the connectors, fasten the camera cable at secure points near the camera and Vision Processor, allowing some slack between the fasteners and the connectors.

For details about the camera and camera cable, see _Camera Reference_.

COGNEX In·Sight

◀ ▲ ▶ Install: External Trigger

A trigger signal connects to the camera, not to the Vision Processor. You attach twisted pair or other suitable wiring through an intermediate, detachable terminal block, which bundles the wires together for easy

assembly and reassembly. Common trigger sources include photocells and proximity switches. Wiring size is 22 to 14 AWG.

RJ-45 connector
(video out)

— Trigger +
— Trigger −
— Strobe +
— Strobe −
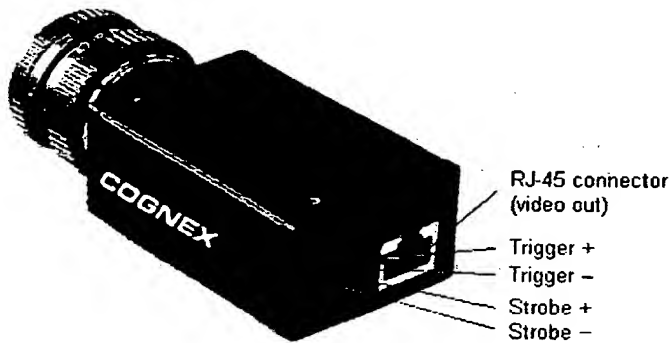
▶ **To Connect an External Trigger**

1. On the 4-position terminal block, insert the Trigger+ and Trigger− lines into the Trigger+ and Trigger− terminals:

Trigger +   Strobe +
   Trigger −   Strobe −
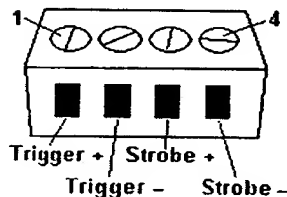
2. Tighten the set screws.
3. If you are using a strobe, connect it to the terminal block as described in *Connecting a Strobe Light*.
4. Remove power from the Vision Processor.
5. Plug the terminal block, which is keyed to prevent incorrect insertion, into the camera.
6. To prevent strain on the connector, fasten the trigger and strobe wires to a secure point near the camera, allowing some slack between the fastener and the terminal block.

For details about the trigger signals, see *Trigger Signal Reference*.

───────────────────────────────────── **COGNEX** In-Sight

◀ ▲ ▶ Install: Strobe Light

The strobe signal fires an external strobe light. It connects to the camera, not to the Vision Processor. You attach twisted pair or other suitable wiring through an intermediate, detachable terminal block, which bundles the wires together for easy assembly and reassembly. Wiring size is 22 to 14 AWG.

◄ ▲ ► Install: Variable Lights

The Vision Processor can control one or two Cognex-compatible variable-intensity lights. Not all applications require dynamic light control, so variable-intensity lights are optional.

To save space on the front panel, the LIGHTS port carries the signals for both lights on one 9-pin connector. Cognex lights therefore include a Y-adapter cable that separates the signals for the two lights. One leg of the Y terminates in a 9-pin DIN connector that plugs into the Vision Processor. The other legs terminate in 7-pin DIN connectors that plug into the lights.
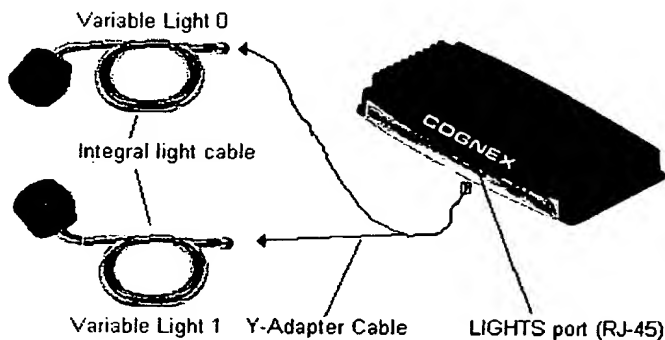
► To Connect Variable-Intensity Lights

NOTE   The connector that carries the light control signals can also supply power to certain Cognex strobe lights. To use a variable-intensity light and a strobe at the same time, you must supply external power to the strobe.

1. Obtain one or two variable-intensity lights, each supplied with a Y-adapter cable (P/N XXX-YYY).
2. Remove power from the Vision Processor.
3. Plug the Y-adapter cable's 9-pin mini-DIN connector into the Vision Processor's LIGHTS port.
4. For the first light, plug one of the Y-adapter's 7-pin DIN connectors into the matching connector on the end of the light's integral cable.
5. If you are using two lights, plug the other light connector into the second light.
6. To prevent strain on the connectors, fasten the cables at secure points near the Vision Processor and lights, allowing some slack between the fasteners and connectors.

For details about variable-intensity lights, see _Available Lights_. For hardware details about the light port, see _Light Control Port_. For software details, see _Controlling Variable Lights_.

COGNEX In-Sight

◄ ▲ ► Install: VGA Monitor

RJ-45 connector
(video out)
Trigger +
Trigger –
Strobe +
Strobe –
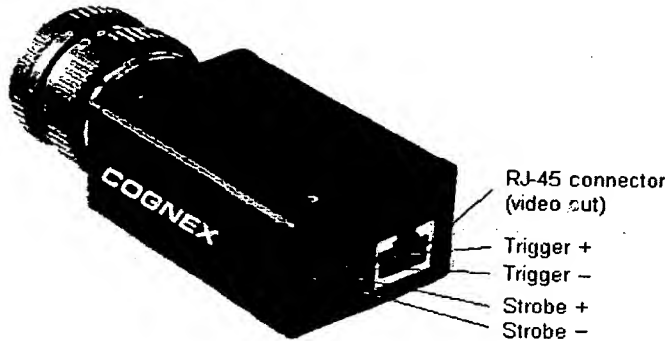
▶ **To Connect Strobe Light**

1. On the 4-position terminal block, insert the Strobe+ and Strobe– lines into the Strobe+ and Strobe– terminals.

Trigger +   Strobe +
Trigger –   Strobe –
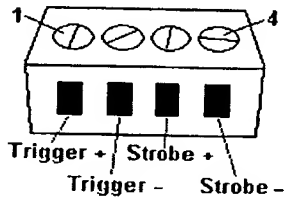
2. Tighten the set screws.
3. Remove the power from the Vision Processor.
4. ??If you are using a Cognex strobe that obtains power from the Vision Processor, plug the 9-pin end of the supplied cable into the matching connector on the front panel of the Vision Processor.

    NOTES
    • The same connector carries the light control signals for variable-intensity lights. To use an variable-intensity light and a strobe at the same time, you must supply external power to the strobe.
    • ??You must set the light power parameter to 255 to enable power to the strobe. For details, see <u>AcquireImage</u>.

5. If you are using an external trigger, connect the trigger wires to the terminal block as described in <u>*Connecting an External Trigger*</u>.
6. Plug the terminal block into the camera.
7. To prevent strain on the connector, fasten the trigger and strobe wires to a secure point near the camera, allowing some slack between the fastener and the terminal block.

For details about the strobe signals, see <u>*Strobe Signal Reference*</u>.

COGNEX In-Sight

Inputs	Circle	A circle, defined by the row and column coordinates for the center point (in
	—Row	pixels from the image origin) and a radius (in pixels).
	—Column	
	—Radius	
	Name	A label for the circle. A string or a reference to one.
	Color	A color, selected from a list.
	Show	ON to enable the circle graphic; OFF to disable it.
Returns	Plot	On success, a Plot structure, which stores the graphic.
Emits	Nothing.	
Errors	Invalid parameter.	
Comments	One way to define the circle graphic in a fixtured coordinate system instead of the image	
	coordinate system is to obtain the circle parameters from a Circle structure.	
See Also	PlotArc, PlotCross, PlotLine, PlotPoint, PlotRegion, PlotString, Getting Started with	
	Image Graphics.	

COGNEX In-Sight

◄ ▲ ► Graphics: PlotCross function

Description	Draws a cross mark graphic with a specified center, angle, height, width, label, and color.
Heading	Graphics/Image
Syntax	PlotCross(Cross, Name, Color, Show)
Inputs	Cross	A cross mark, defined by its center point (row and column, in pixels from
	—Row	the image origin), its angle (a rotation, ± 360° CCW from the image row
	—Column	axis), and its height and width (in pixels).
	—Angle	
	—High	
	—Wide	
	Name	A text label for the cross mark. A string or a reference to one.
	Color	A color, selected from a list.
	Show	ON to enable the cross mark; OFF to disable it.
Returns	Plot	On success, a Plot structure, which stores the graphic.
Emits	Nothing.	
Errors	Invalid parameter.	
Comments	One way to define the cross mark graphic in a fixtured coordinate system instead of the	
	image coordinate system is to obtain the cross parameters from a Cross structure.	
See Also	PlotArc, PlotCircle, PlotLine, PlotPoint, PlotRegion, PlotString, Getting Started Plotting	
	Graphics.	

COGNEX In-Sight

◄ ▲ ► Graphics: PlotLine function

Description	Draws a line graphic between two points.
Heading	Graphics/Image
Syntax	PlotLine(Line, Name, Color, Show)
Inputs	**Line** A line, defined by its end points, in pixels from the image origin.
	—Row 0
	—Column 0
	—Row 1
	—Column 1
	Name A text label for the line graphic. A string or a reference to one.
	Color A color, selected from a list.
	Show ON to enable the line graphic; OFF to disable it.
Returns	**Plot** On success, a Plot structure, which stores the graphic.
Emits	Nothing.
Errors	Invalid parameter.
Comments	One way to define the line graphic in a fixtured coordinate system instead of the image coordinate system is to obtain the line parameters from a Line structure.
See Also	PlotArc, PlotCircle, PlotCross, PlotPoint, PlotRegion, PlotString, *Getting Started Plotting Graphics*.

COGNEX In Sight

◄ ▲ ► Graphics: PlotPoint function

Description	Draws a point graphic at a specified point in the image.
Heading	Graphics/Image
Syntax	PlotPoint(Point, Name, Color, Show)
Inputs	**Point** A point, defined by its row and column coordinates, in pixels from the
	—Row image origin.
	—Column
	Name A label for the point. A string or a reference to one.
	Color A color, selected from a list.
	Show ON to enable the point graphic; OFF to disable it.
Returns	**Plot** On success, a Plot structure, which stores the graphic.
Emits	Nothing.
Errors	Invalid parameter.
Comments	One way to define the point graphic in a fixtured coordinate system instead of the image coordinate system is to obtain the row and column parameters from a Point structure.
See Also	PlotArc, PlotCircle, PlotCross, PlotLine, PlotRegion, PlotString, *Getting Started Plotting Graphics*.

COGNEX In Sight

◄ ▲ ► Graphics: PlotRegion function

Description	Draws a rectangular region graphic, potentially rotated and curved.	
Heading	Graphics/Image	
Syntax	PlotRegion(Region, Name, Color, Show)	
Inputs	**Region**	A rectangular region, defined by its top-left point (row and column, in
	—Row	pixels from the image origin), its angle (a rotation, ± 360° CCW from the
	—Column	image row axis), its height and width (in pixels), and its curvature (the angle
	—Angle	between the sides of the region, ± 360° CCW). For curved regions,
	—High	"width" is the arc distance at the center of the region.
	—Wide	
	—Curve	
	Name	A text label for the region graphic. A string or a reference to one.
	Color	A color, selected from a list.
	Show	ON to enable the region graphic; OFF to disable it.
Returns	**Plot**	On success, a Plot structure, which stores the graphic.
Emits	Nothing.	
Errors	Invalid parameter.	
Comments	One way to define the region graphic in a fixtured coordinate system instead of the image coordinate system is to obtain the region parameters from a Region structure.	
See Also	PlotArc, PlotCircle, PlotCross, PlotLine, PlotPoint, PlotString, *Getting Started Plotting Graphics*.	
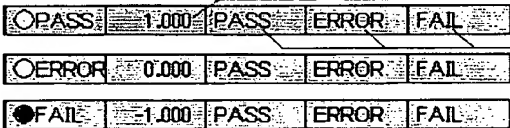
COGNEX In-Sight

◄ ▲ ► Graphics: PlotString function

Description	Draws a string graphic at a specified point, in a specified color.	
Heading	Graphics/Image	
Syntax	PlotString(String, Point, Color, Show)	
Inputs	**String**	A string or a reference to one.
	Point	The location of the string, defined by its top-left point, in pixels from the
	—Row	image origin.
	—Column	
	Color	A color, selected from a list.
	Show	ON to enable the string graphic; OFF to disable it.
Returns	**Plot**	On success, a Plot structure, which stores the graphic.
Emits	Nothing.	
Errors	Invalid parameter.	
Comments	None.	
See Also	PlotArc, PlotCircle, PlotCross, PlotLine, PlotPoint, PlotRegion, *Getting Started Plotting Graphics*.	

COGNEX InSight

◄ ▲ ► Graphics: Status function

Description	Puts a status display in a cell. Green if positive; yellow if 0.00; red if negative.
Heading	Graphics/Worksheet
Syntax	Status(Status, GreenString, YellowString, RedString)

Inputs	**Status**	The cell monitored for its status. Green if **Status** is positive; yellow if 0.0; red if negative or an error.
	GreenString	A label for the status graphic when **Status** is positive. A string or a reference to one.
	YellowString	A label for the status graphic when **Status** is zero. A string or a reference to one.
	RedString	A label for the status graphic when **Status** is negative or error. A string or a reference to one.

Returns	0.0 on success.
Emits	Nothing.
Example	

Status parameter, by reference
String parameter, by reference

Errors	Invalid parameter.
Comments	None.
See Also	Chart, ColorLabel, *Getting Started with Worksheet Graphics*.

COGNEX InSight

◄ ▲ ► Hist: Overview

Histogram analysis is a technique for finding the distribution of grayscale values in an image. Preparing a histogram involves countin the number of occurrences of each grayscale value, placing each total in a "bin." An 8-bit image has 256 grayscale values, from 0 (black) to 255 (white), so its histogram has 256 bins. The bin number equals the grayscale value totaled there. Bin 0, for instance, holds the count of pixels at grayscale 0; bin 1 holds the count at grayscale 1; and so on.

Common uses for histogram analysis are feature extraction and image analysis. For example, suppose your application inspects industrial parts that normally yield histograms with a peak near grayscale 64 (the part) and another near grayscale 200 (the background). Also suppose that you occasionally acquire blank images, with all values near grayscale 200. Testing for images that lack the peak near grayscale 64 would let you exclude the blank images from further processing. Here, the histogram acts like a light meter.

This section introduces In-Sight's Histogram functions, discussing the following topics:

- Getting Started with Histograms
- Understanding Histograms
- Histogram Statistics
- Histogram Reference.

With In-Sight, you generate a histogram with the ExtractHistogram function (under Vision Processing), which stores the histogram and its statistical profile in a Hist structure. Values emitted to the worksheet include an automatically computed binary threshold (the grayscale value that best separates the image into light and dark regions) and the averages for the light region, dark region, and entire histogram. Statistics available for all or part of the histogram include the number of pixels, the first and last non-zero gray level (head and tail), the most and least common gray levels (maximum and minimum), the sum, and the standard deviation.

COGNEX In·Sight

◄ ▲ ► Hist: Getting Started

This topic describes basic steps for extracting a histogram from an image. The result looks like this:

Region of interest

Threshold

Maximum value

Theshold value

Histogram

Minimum value ⎯⎯
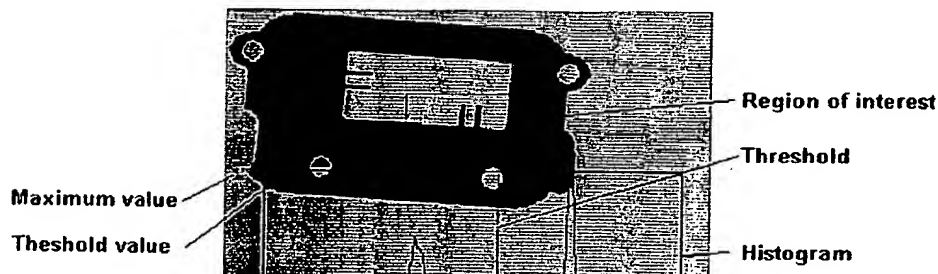
▶ **To Extract a Histogram**

1. Acquire an appropriate image. For this exercise:
   a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
   b. Click △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
   c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

   To acquire a new image at any time, hold □ and click ✕.

2. Leaving cell **A1** empty, move to cell **A2** and click ✕ to open the Formula Builder. From the **Vision Processing** category, select **Histogram**, and click ✕ to open the ExtractHistogram property sheet. NOTE Try to leave a blank row above functions that emit formulas. In-Sight prints descriptive labels above emitted formulas if it has room for the labels. If the row above is occupied, it emits only the numeric values, without labels.

3. To define a region of interest, select the **Region** head and click ✕. In-Sight hides the property sheet and displays the Region cursor, which you interactively adjust with the Control Pad. Outline an area that contains a feature showing a range of grayscale values.

4. When satisfied with the region, click ✕ to accept it, and then click **Run**. ExtractHistogram() computes a histogram for the specified region and displays it at the bottom of the property sheet. To view only the histogram and image, click △ to toggle the property sheet overlay ON or OFF:

   Region of interest
   Threshold
   Maximum value
   Theshold value
   Histogram
   Minimum value
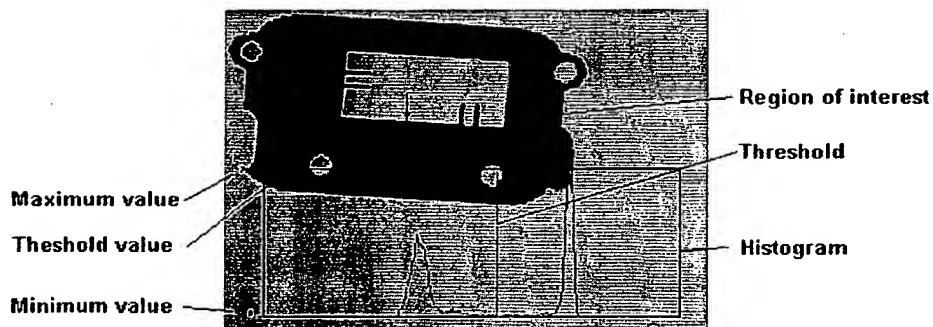
5. Click **OK**. ExtractHistogram() recomputes the histogram, stores it in a Hist structure, and emits a block formulas that put statistical results on the worksheet:

	Thresh	Contrast	AveDark	AveBright	Average
⑤Hist	81.000	81.984	38.721	120.706	86.330

   See *Histogram Statistics* for details about the emitted values.

6. To use an emitted value in a formula, create a reference to its cell. To simplify the worksheet, consider deleting unneeded emitted formula.

By default, In-Sight displays the histogram graphic only when you highlight the Hist structure, as described in *Flyover Graphics*. To display the histogram graphic all the time, set the ExtractHistogram() function's **Show** parameter to **Show All**.

COGNEX In-Sight

◄ ▲ ► Hist: Understanding

A *histogram* is an array of integers that represent the distribution of the pixel values in a region of interest. Each element in the array, called a *bin*, holds a count of the number of pixels at a particular grayscale value. The bin indexes correspond to the grayscale values counted there. That is, bin 0 holds the number of pixels at grayscale 0, bin 1 holds the number of pixels at grayscale 1, and so on. The total of all bin counts always equals the number of pixels in the region.

A typical histogram has *peaks*, or *modes*, corresponding to the pixel values in the dominant features in the image. For example, consider a binary image and its histogram:
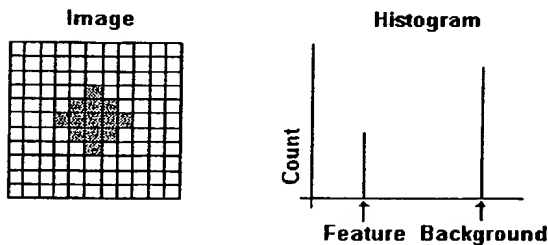
Here, only two bins have non-zero contents: the pixel value of the feature and the pixel value of the background.

Real images seldom have histograms like this. Noise from various sources—irregular lighting, uneven printing, electrical noise, spatial quantization error, and so on—combine to spread out the peaks. A more realistic histogram of the scene as viewed through a camera might look like this:
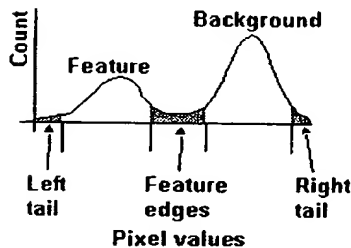
This histogram clearly shows both peaks, which have spread out to include neighboring pixel values. Their

relative proportions nonetheless remain about the same as those in the "ideal" (binary) histogram. The less-populated pixel values between the two principal peaks are the *edges* of the feature, which are neither wholly dark nor wholly light. The left and right *tails* contain outlying points, having values that might be unreliable due to noise. To limit the effect of noise, your application might ignore the tails as unreliable end points.

COGNEX In Sight

◄ ▲ ► Hist: Statistics

The ExtractHistogram function computes a histogram and stores it in a Hist structure. It also stores a set of statistical values describing the histogram. Some of the statistical values are emitted to the worksheet but most are not. The emitted values are:

- **Thresh**. Binary threshold, which is the grayscale value that best separates the histogram into dark and bright regions—the optimum bimodal distribution. Emitted by HistThresh.
- **Contrast**. Grayscale contrast, which is the difference between the average bright value and the average dark value (that is, **AveBright** minus **AveDark**). Emitted by HistContrast.
- **AveDark**. The average grayscale value in the dark region (the values below the threshold). Computed by HistMean over the range of 0 through **Thresh − 1**.
- **AveBright**. The average grayscale value in the bright region (the values above the threshold). Computed by HistMean over the range of **Thresh** through 255.
- **Average**. Arithmetic average over the histogram as a whole, from bin 0 to bin 255. Emitted by HistMean.

Additional values, stored but not emitted, that can be read through Data Acess functions are:

- **Count**. Number of pixels. Available from HistCount.
- **Head**. First non-zero gray level. Available from HistHead.
- **Tail**. Last non-zero gray level. Available from HistTail.
- **Maximum**. Most-common (modal) gray level. Available from HistMax.
- **Minimum**. Least-common gray level. Available from HistMin.
- **SDev**. Standard deviation. Available from HistSDev.
- **Sum**. Sum of the grayscale values. Available from HistSum.
- **SumSquare**. Sum of the squares of the grayscale values. Available from HistSumSquare.
- **Value**. Number of values in a bin or range of bins. Available from GetValue.

You can read any of these values over the histogram as a whole (the default) or limited to a specified grayscale range.

COGNEX In Sight

◄ ▲ ► Hist: Function Reference

ExtractHistogram Computes a grayscale histogram from an image region, storing it in a Hist structure. Emits histogram statistics.

Hist structure	Stores a histogram and histogram statistics. Generated by ExtractHistogram().
HistContrast	Returns the grayscale contrast value (the average bright value minus the average dark value) for all or part of the histogram.
HistCount	Returns the number of pixels, by default for the entire histogram.
HistHead	Returns the index of the first non-zero gray level for all or part of the histogram.
HistMax	Returns the most-common (modal) gray value for all or part of the histogram.
HistMean	Returns the average grayscale value for all or part of the histogram.
HistMin	Returns the least-common grayscale value for all or part of the histogram.
HistSDev	Returns the standard deviation value for all or part of the histogram.
HistSum	Returns the sum of the grayscale values for all or part of the histogram.
HistSumSquare	Returns the sum-of-squares value for all or part of the histogram.
HistTail	Returns the index of the last non-zero gray level for all or part of the histogram.
HistThresh	Returns the optimum binary threshold value (the grayscale value that best separates the histogram into dark and bright regions).

A Data Access function that get value from a Hist structure is GetValue.

COGNEX In·Sight

◄ ▲ ► Hist: ExtractHistogram function

Description	Computes a grayscale histogram from an image region, storing it in a Hist structure. Emits histogram statistics.
Heading	Vision Processing/Histogram
Syntax	ExtractHistogram(Image, Fixture, Region, Show)
Inputs	**Image** Image data source, a reference to an Image structure. Default is A0.
	Fixture Fixture origin, defined by row, column, and angle offsets, typically to
	—**Row** compensate for image-to-image variation in position, orientation, or both.
	—**Column** **Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—**Theta** rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Region Region of interest. **X** and **Y** define the top-left point by offsets from the
	—**X** Fixture origin, in pixels. **High** and **Wide** are the height and width, in pixels.
	—**Y** **Angle** is a rotation from the Fixture X axis, ± 360° CCW. See *Defining a*
	—**High** *Region of Interest*.
	—**Wide**
	—**Angle**
	Show Selects the graphics to display, as described in *Flyover Graphics*:

Appendix I - Page **211**

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Hist**	On success, a <u>Hist structure</u>, which stores the histogram.
Emits	**Thresh**	The gray level that best separates the histogram into dark and bright regions—the optimum bimodal distribution. Emitted by <u>HistThresh</u>.
	Contrast	The difference between the average bright value and the average dark value (**AveBright** minus **AveDark**). Emitted by <u>HistContrast</u>.
	AveDark	The average grayscale value in the dark region (values below the threshold). Computed by <u>HistMean</u> from 0 to **Thresh** − 1.
	AveBright	The average grayscale value in the dark region (values above the threshold). Computed by <u>HistMean</u> from **Thresh** to 255.
	Average	The average value of the histogram as a whole, from bin 0 to bin 255. Emitted by <u>HistMean</u>.
Errors	Invalid parameter.	
Comments	None.	
See Also	<u>HistContrast</u>, <u>HistCount</u>, <u>HistHead</u>, <u>HistMax</u>, <u>HistMin</u>, <u>HistSum</u>, <u>HistSumSquare</u>, <u>HistTail</u>, <u>HistThresh</u>, <u>GetValue</u>, *Getting Started with Histograms*	

COGNEX In-Sight

◄ ▲ ► Hist: Hist structure

Description   Stores a histogram and histogram statistics. Generated by ExtractHistogram().

Values

Value	Description	Access function
Contrast	Grayscale contrast. Equal to the average bright value (>= Thresh) minus the average dark value (< Thresh). Emitted for the entire histogram, bin 0 to bin 255, but optionally limited to a range of bins.	<u>HistContrast</u>
Count	Number of pixels in a bin or range of bins. Not emitted. Defaults to the entire histogram, bin 0 through bin 255.	<u>HistCount</u>
Head	First non-zero gray level. Not emitted. Defaults to the entire histogram but optionally limited to a range of bins.	<u>HistHead</u>
Maximum	Most-common gray level (the statistical mode). Not emitted. Defaults to the entire histogram but optionally limited to a range of bins.	<u>HistMax</u>
Mean	The arithmetical average of the histogram, optionally limited to a range of bins. Separately emitted for the bright region, dark region, and entire histogram.	<u>HistMean</u>
Minimum	Least-common gray level. Not emitted. Defaults to the entire histogram but optionally limited to a range of bins.	<u>HistMin</u>

Appendix I - Page **212**

	SDev	Standard deviation. Not emitted. Defaults to the entire histogram but optionally limited to a range of bins.	HistSDev
	Sum	Sum of the grayscale values in a bin or range of bins. Not emitted. ??Defaults to the entire histogram??	HistSum
	SumSquare	Sum of the squares of the grayscale values. *Not* square-rooted, since you can separately calculate the square root of the sum-square value if desired. Not emitted. Defaults to the entire histogram but optionally limited to a range of bins.	HistSumSquare
	Tail	Last non-zero gray level. Not emitted. Defaults to the entire histogram but optionally limited to a range of bins.	HistTail
	Threshold	The gray level that best binarizes the image. Emitted for the entire histogram, bin 0 to bin 255, but optionally limited to a range of bins.	HistThresh
	Value	Number of values in a specified bin. ??Which bin is the default?? Not emitted.	GetValue
Comments	For most histogram statistics, you can obtain values for the entire histogram (the default) or limited to a specified grayscale range (by supplying the access function's range start and range end parameters).		
See Also	ExtractHistogram.		

COGNEX In·Sight

◄ ▲ ► Hist: HistContrast function

Description	Returns the grayscale contrast value (the average bright value minus the average dark value) for all or part of the histogram.
Heading	Vision Processing/Histogram
Syntax	HistContrast(Hist, [Range Start, Range End])
Inputs	**Hist** A reference to a Hist structure, generated by ExtractHistogram.
	Range A range of bins, defined by starting and ending grey levels (0 through 255,
	—Start inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End histogram).
Returns	Grayscale contrast, a floating-point value.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	Hist structure, HistCount, HistHead, HistMax, HistSDev, HistTail.

COGNEX In·Sight

Appendix I - Page **213**

◄ ▲ ► Hist: HistCount function

Description	Returns the number of pixels in a range of bins.
Heading	Vision Processing/Histogram
Syntax	HistCount(Hist, [FirstBin, LastBin])
Inputs	**Hist** A reference to a <u>Hist</u> structure.
	Range A range of bins, defined by starting and ending grey levels (0 through 255,
	—Start inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End histogram).
Returns	The number of pixels in a range of gray levels.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	<u>ExtractHistogram</u>.

COGNEX In-Sight

◄ ▲ ► Hist: HistHead function

Description	Returns the index of the first non-zero gray level, by default for the entire histogram.
Heading	Vision Processing/Histogram
Syntax	HistHead(Hist, [Range Start, Range End])
Inputs	**Hist** A reference to a <u>Hist structure</u>, generated by <u>ExtractHistogram</u>.
	Range A range of bins, defined by starting and ending gray levels (0 through 255,
	—Start inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End histogram).
Returns	Index of the first non-zero gray level. Valid range, 0 through 255.
Emits	Nothing.
Errors	Invalid parameter.
Comments	Specifying a range lets you separately obtain the local "heads" from histograms with more than one peak.
See Also	<u>Hist structure</u>, <u>HistContrast</u>, <u>HistCount</u>, <u>HistMax</u>, <u>HistSDev</u>, <u>HistTail</u>.

COGNEX In-Sight

◄ ▲ ► Hist: HistMax function

Appendix I - Page **214**

Description	Returns the most-common (modal) gray value, by default for the entire histogram.
Heading	Vision Processing/Histogram
Syntax	HistMax(Hist, [FirstBin, LastBin])
Inputs	**Hist** A reference to a <u>Hist</u> structure.
	Range A range of bins, defined by starting and ending grey levels (0 through 255,
	—Start inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End histogram).
Returns	The most common (modal) gray level within a range.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	<u>ExtractHistogram</u>, <u>HistMin</u>.

COGNEX In-Sight

◄ ▲ ► Hist: GetMean function

Description	Returns the average grayscale value, by default for the entire histogram.
Heading	Vision Processing/Histogram
Syntax	GetMean(Hist, [FirstBin, LastBin])
Inputs	**Hist** A reference to a <u>Hist structure</u>, generated by ExtractHistogram().
	Range A range of bins, defined by starting and ending grey levels (0 through 255,
	—Start inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End histogram).
Returns	The mean value of the buffer.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	<u>ExtractHistogram</u>, <u>Hist structure</u>.

COGNEX In-Sight

◄ ▲ ► Hist: HistMin function

Description	Returns the least-common grayscale value, by default for the entire histogram.
Heading	Vision Processing/Histogram
Syntax	HistMin(Hist, [FirstBin, LastBin])

Inputs	Hist	A reference to a <u>ExtractHistogram</u> structure.
	Range	A range of bins, defined by starting and ending grey levels (0 through 255,
	—Start	inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End	histogram).
Returns		The least-common value in the range.
Emits		Nothing.
Errors		Invalid parameter.
Comments		None.
See Also		<u>HistMax</u>.

COGNEX In·Sight

◄ ▲ ► Hist: HistSDev function

Description	Returns the standard deviation value, by default for the entire histogram.
Heading	Vision Processing/Histogram
Syntax	HistSDev(Hist, [FirstBin, LastBin])

Inputs	Hist	A reference to a <u>Hist structure</u>, generated by ExtractHistogram().
	Range	A range of bins, defined by starting and ending grey levels (0 through 255,
	—Start	inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End	histogram).
Returns		Standard deviation, a standard statistical measure.
Emits		Nothing.
Errors		Invalid parameter.
Comments		None.
See Also		<u>ExtractHistogram</u>, <u>Hist structure</u>.

COGNEX In·Sight

◄ ▲ ► Hist: HistSum function

Description	Returns the sum of the grayscale values, by default for the entire histogram.
Heading	Vision Processing/Histogram
Syntax	HistSum(Hist, [FirstBin, LastBin])

Inputs	Hist	A reference to a <u>Hist</u> structure.
	Range	A range of bins, defined by starting and ending grey levels (0 through 255,
	—Start	inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End	histogram).
Returns		The total number of pixels in the range.
Emits		Nothing.
Errors		Invalid parameter.
Comments		None.

See Also   ExtractHistogram.

COGNEX In-Sight

◄ ▲ ► Hist: HistSumSquare function

Description	Returns the sum-square value for all or part of the histogram.
Heading	Vision Processing/Histogram
Syntax	HistSumSquare(Hist, [Range Start, Range End])
Inputs	**Hist** A reference to Hist structure, generated by ExtractHistogram.
	Range A range of bins, defined by starting and ending grey levels (0 through 255,
	—**Start** inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—**End** histogram).
Returns	Sum-square value.
Emits	Nothing.
Errors	Invalid parameter.
Comments	The result is the simple sum of the squares, not the square root of the sum of the squares.
See Also	Hist structure, HistContrast, HistCount, HistHead, HistMax, HistSDev, HistTail.

COGNEX In-Sight

◄ ▲ ► Hist: HistTail function

Description	Returns the index of the last non-zero gray level, by default for the entire histogram.
Heading	Vision Processing/Histogram
Syntax	HistTail(Hist, [Range Start, Range End])
Inputs	**Hist** A reference to a Hist structure.
	Range A range of bins, defined by starting and ending grey levels (0 through 255,
	—**Start** inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—**End** histogram).
Returns	The index of the last non-zero gray level. Valid range, 0 through 255.
Emits	Nothing.
Errors	Invalid parameter.
Comments	Specifying a range lets you separately obtain the local "tails" from histograms with more than one peak.
See Also	Hist structure, HistContrast, HistCount, HistHead, HistMax, HistSDev.

COGNEX In-Sight

◄ ▲ ► Hist: HistThresh function

Description	Returns the optimum binary threshold value (the grayscale value that best separates the histogram into dark and bright regions).	
Heading	Vision Processing/Histogram	
Syntax	HistThresh(Hist, [FirstBin, LastBin])	
Inputs	**Hist**	A reference to a <u>Hist</u> structure, generated by <u>ExtractHistogram</u>.
	Range	A range of bins, defined by starting and ending grey levels (0 through 255,
	—Start	inclusive). Start defaults to 0, and End defaults to 255 (the entire
	—End	histogram).
Returns	Threshold.	
Emits	Nothing.	
Errors	Invalid parameter.	
Comments	None.	
See Also	<u>Hist structure</u>, <u>HistContrast</u>, <u>HistCount</u>, <u>HistHead</u>, <u>HistMax</u>, <u>HistSDev</u>, <u>HistTail</u>.	

COGNEX In-Sight

◄ ▲ ► Image Processing: Overview

The Image Processing heading, under <u>Vision Processing</u>, offers functions that change the values of the pixels in an image, yielding a modified image as a result. Its subheadings are:

Image Processing   Performs pixel-processing or image enhancement operations:

- Neighborhood filters operations consider the values of the surrounding pixels when changing a pixel value
- Point filters ignore the surrounding pixels.

Many In-Sight applications use image processing to enhance an image before applying <u>feature extraction</u> to obtain information about its content.

Image Comparison   Compares an image with a template to detect their differences. A common use is detecting anomalies in inspection applications.

Image Warping   A polar transformation that unwraps a curved or circular feature into a second, rectangular image. (<u>TransWarpToPixel</u>, under Coordinate Transforms, maps a point in the warped image back to the original image.)

This section describes image comparison, image processing, and image warping:

- <u>Gettin Started with Image Comparison</u>
- <u>Understanding Image Comparison</u>
- <u>Getting Started with Image Processing</u>
- <u>Understanding Image Processing</u>
- <u>Getting Started with Image Warping</u>
- <u>Image Processing Reference</u>.

━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX** In-Sight

◄ ▲ ► Image Comparision: Getting Started

You typically use normalized Template Subtraction to determine presence/absence, find defects, and locate identifying markers. In this section, you use template comparison to compare an image with a template to measure the degree of disagreement.

► **To Compare an Image**

1. **Gather Training Images.** Provide the tool a number of training images. You do this by setting up the ROI, fixtured, and setting the AddTrainImage flag to 1. Every time the job runs, a new image is added.
2. **Set the AddTrainImage flag to 0.**
3. **Set Up Training.** Select the normalization method: graylevel scaling, histogram fitting, mean/std dev fitting or image fitting. None can also be used.
4. Decide whether to remove extreme gray values from the calculation. The tails of the image graylevel distribution are sometimes the result of noise and can be unreliable. Set the Tail Clipping flag to true and the amounts to clip off either end of the distribution in the Left Tail and Right Tail variables.

5. **Set the Edge Tolerance.** This adjusts training for small differences in registration that may not show up in the training set.
6. **Train the tool.** Set the Train flag to 1.
7. **Exit the tool.** This trains, creating an ideal image of averages, and collects pixel by pixel statistics - the standard deviation of the training set.
8. Set up for execution. Set the Train flag to 0.
9. Set the Tolerance Percent. This is the range of variation expected. 2 is the best — 2 standard deviations.
10. Set the noise tolerance, to allow for noise in regions of little variation in the training images.
11. Set up post processing. The tool generates a black and white image, where differences show up as white blobs. Use the blob tool to detect them.

COGNEX In-Sight

◄ ▲ ► Image Comparison: Understanding

Some machine vision applications look for differences between an ideal part and a test part. Changes stand out when the image of the ideal part is subtracted from the image of the test part. Applications that can benefit from image comparison include:

- Defect detection
- Sorting similar
- Searching for identifying markers.

In addition, the performance of many vision tools can be enhanced by subtracting known backgrounds from an image.

Here are some key concepts in image comparison:

1. The definition of image comparison
2. Image sizes
3. Values of result pixels
4. The normalization of gray levels in images.

Image Comparison

When the comparison tool compares one image (called the *template image*) to another image (the *input image*), it subtracts the value of each pixel in the template image from the corresponding pixel in the input image. It then compares the result to the tolerance value for that pixel, stored in a third image (the sigma or standard deviation image), and stores the result in a fourth image, the output image. The result image pixels are white where the subtraction results are outside the tolerance value, and black where they are within the tolerance. The output image can be used as input to subsequent tools, such as the blob tool or search tool, or it can be processed further using the process tool.

Image Sizes

If the template image and the input region in the input image differ in size, the compare tool cannot make a meaningful comparison, as it would have to align the images and region in some manner. The size of the template image is set when the compare tool is trained, as the size of that input region. The input region height and width should not be altered unless you want to retrain the template image and sigma image.

If you define the Compare tool input region with respect to a fixture that rotates each time the spreadsheet tool runs, that region also will be rotated. Although the region rotates, the input image will be the same size as the template image, and the compare filter will execute. If you define the Compare tool input region by referencing a height and width which is calculated elsewhere in the spreadsheet, the Compare tool will retrain every time the input region height or width changes.

Pixel Values

The Comparison tool uses one method for determining output. It checks both positive and negative differences differences between the template and input images, and returns white pixels where the differences are beyond the tolerated range in either the positive or negative direction. Because it captures both negative and positive changes, it does not allow you to distinguish between lighter and darker areas. The pixel values in the returned image are either 0 (black) or 255 (white).

Image Normalization

The process of image subtraction is sensitive to changes in lighting, jitter, and noise. Even a small variation in the ambient light level can result in a faint copy of the input image in the output image. Slight inaccuracies in fixturing can introduce jitter which produces artifacts along edges. And noise can result from imperfections in the acquisition, optics, or from the surface of the object being imaged.

You can compensate for these problems during setup and at runtime. During setup, there are three ways to adjust the response of the comparison tool:

- Normalizing the template and input images
- Remove extreme values from the image, since these are likely to be noise
- Building edge tolerance into the template images.

At runtime, there are two ways to compensate for variations:

- Setting the absolute level of noise tolerated
- Setting the spread of variation tolerated.

Take a common histogram of an image where lighting has been adjusted correctly. The histogram has two peaks, representing the features of interest and the background. The mean is the average value of pixels in this histogram, and the standard deviation is the overall spread of the pixel values. The range of the grayscale values goes from 0 (dark) to 255 (bright). The histogram of an image is used to normalize its grayscale values, both of the input and the template image, so that the comparisons are less susceptible to noise and image lighting variations. There are three different methods available, from fastest to slowest:

- **Greylevel Scaling** (or range stretching): adjusts the pixel values of an image to cover the a given range of values. The Compare tool uses full grayscale range.

Appendix I - Page **221**

- **Mean/StdDev Fitting**: Matching the means and standard deviations of the template and input image histograms. This method aligns the mean pixel values, and stretches or shrinks the range of the input image grayscales to match the range of the template image, as measured by the standard deviation of the histogram.
- **Histogram Fitting**: Calculates an offset and scaling factor which minimizes the difference between the curve of the template image histogram and the input image histogram, and therefore a minimum difference between their pixel value distributions. This is the slowest method, but the least susceptible to noise and artifacts.

Typically, the extreme left and right tails of such a histogram are susceptible to noise and varying lighting conditions, and are unreliable. Before normalizing image grayscales, it is sometimes useful to remove those values. The Exclude Extremes or Tail Clipping parameter provides this ability.

The *left tail* of the histogram is defined as that gray value to the left of which a user-defined percentage of pixels fall (dark pixels), and the *right tail* of the histogram is defined as that gray value to the right of which a user-defined percentage of pixels fall (bright pixels). Typically, you choose the percentage of pixels for each tail such that the histogram shape between the left and right tails is consistent each time the tool is run.

Typically, the edges of the object in the template image and in the input image do not exactly align, either because of inaccuracy in the fixturing or because of variation in the object being imaged.

The Edge Tolerance parameter allows you to set the degree of tolerance in the edge position during training. The Edge Tolerance parameter sets the number of fractional pixels by which the edge can vary. During training, the compare tool detects edges in the image used to build the template image, and builds the variation into the template image (actually, in the sigma image) where the edges occur.

At runtime, the input image will have natural low-level variation in the pixel values within an area of constant color. This results from noise in the hardware, small variations in lighting, and the crudeness of the camera optics, among other things. This variation is captured partially during the creation of the template and sigma images, and is expressed as tolerance in the sigma image. In practice, that capture is imperfect, and it's necessary to provide two methods for adjusting the tolerance. First, you can set an absolute minimum level of difference to tolerate. The Noise Tolerance parameter (currently named Basic Tolerance), is the number of graylevels of difference to always tolerate.

Second, you can set how tightly the variation in the training image is applied (where variation is the sigma image). The sigma image stores the standard deviation of the neighborhood of each pixel. The Variance Tolerance parameter determines how many standard deviations of variation to use. The default, two standard deviations, captures 90% of the range of variance, omitting the upper and lower ends of the variance. This is useful if the grayscale changes due to lighting variations or object orientation are not uniform, and therefore not fully correctable by the global grayscale normalization.

Setting up the Compare Tool

1. Fixture the tool. The compare tool will not work effectively unless either it is fixtured, or it operates on a transformed input image which is itself fixtured.
2. On a training image, set the region to correspond to the part of the input image you wish to use as a template.
3. Choose a Normalization method. Choosing no method will simply do an arithmetic comparison of the

two images without normalizing the gray levels first.

4. Decide whether or not to clip off outlying pixel values. It can be helpful to histogram the input region to see the distribution of values. Set the Exclude Extremes parameter and clipping percentages accordingly.

5. Set the Edge Tolerance parameter, based on your estimation of the reliability of preciseness of the fixturing and the tolerance of the application domain for things that are slightly out of place.

6. When you exit the tool, it will retrain. This can take several seconds if the template region is large.

7. Test it by running the job a few times, and moving the object under the camera, to see if the fixturing is robust enough. You can adjust the Variance Tolerance and Basic Tolerance (Noise Tolerance) parameters without retraining the template, if the trained template and chosen normalization algorithm do not seem to be effective enough (you are seeing large areas reported as defective).

8. Run the output image through postprocessing to eliminate noise. The image may have salt-and-pepper speckles, or faint outlines where there are edges. For many applications, you will want to run an Open with a small kernel, to get rid of the noise.

9. Run the output image of the postprocessing through blob analysis. Ideally, there should be no blobs, or blobs where you want to find them.

Applications

- Defect detection, to identify products which vary from an ideal product.

  Use the above setup, with the defects showing up as blobs. This will locate missing or misaligned parts.
- Defect detection, to identify damage during processing.

  Set up two jobs to run in sequence. In the first job, an image of the part is acquired before processing, and is used as a training image for the Compare tool (the job runs Train for the tool). The second job runs after the part returns from the manufacturing process, and runs as a normal template compare. Defects introduced will show up as white blobs. One example of this is wafer production processes which may chip the wafer.
- Searching for identifying marks (where those marks are the only thing that changes from instance to instance of the imaged object). The training image must be an unmarked instance of the object. The location of the mark will show up as a blob in the output of Compare.
- Sorting similar objects: can be compared to an ideal 'reference' object with Compare. Their differences from the ideal will show up as white regions in the output image. Those white regions can be measured by the edge or blob tools, and the results used as a basis for sorting.

═══ **COGNEX In Sight**

◄ ▲ ► **Image Processing: Getting Started**

In this topic, you define a region of interest, then apply several processing operations to it? this section could flow like this?

► **To Process an Image**

1. Acquire an appropriate image. For this exercise:
    a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
    b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
    c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

    To acquire a new image, hold ☐ and click ✕.

2. Open ???
3. Apply a low pass filter to the original image
4. store th

=== **COGNEX** In-Sight

◄ ▲ ► Image Processing: Understanding

??Drop this topic?? To process an image, you define an ROI for processing, typically by offsets from the Fixture row, column, and theta values. You also define a script-like sequence of processing operations to be applied within ROI. This section describes the script operations in groups:

- Image Processing Filters.

The constant-to-image operations apply a fixed integer value to the ROI; the image-to-image operations apply a second pixel value from a buffered image; and the filter operations are neighborhood operations.

=== **COGNEX** In-Sight

◄ ▲ ► Image Processing: Filters

A filter is an image processing operation that considers the values of surrounding pixels when changing a pixel value. The region considered during processing is called the *neighborhood* or *kernel*. For In-Sight, the kernel is always rectangular and controlled by a height and width value. If the kernel has an odd height and an odd width, it has a unique center. If it has an even height or width, then ?what happens?. ?For In-Sight, all pixels inside the kernel count equally toward the final result. In-Sight therefore offers the Erode and Dilate filters but not a Gaussian blur, which the kernel in a normal distribution.?

Filters differ in the arithmetical operation applied to the center pixel. A low pass filter, for example, changes the center pixel to the kernal mean, but an erode filter changes it to the kernal minimum. In-Sight offers the

filters that are generally useful and practical.

?names are based on white features on a dark background?

LoPass and HiPass Filters

LoPass and HiPass are related neighborhood-averaging filters. With LoPass, the new center value is the average (the arithmetic mean) of the values inside the kernel. LoPass consequently tends to ??. With HiPass, XXX. ?spatial only—light and dark make no diff?

- LoPass passes low-frequency image data and attenuates high-frequency data, much like the bass control on a stereo. Useful for eliminating dust, scratches and other distractacting defects. The underlying algorithm is a neighborhood-averaging filter that sets the pixel at the center of the kernel to the mean within the kernel.
- HiPass passes high frequency image data and attenuates low-frequency data, much like the treble control on a stereo. The underlying algorithm sets the pixel at the center of the kernel to the kernel mean minus the original value. (The kernel mean is the LoPass result.). The following figure shows before-and-after results for HighPas and LoPass:

Erode and Dilate Filters

Erosion is a neighborhood minimum operation. The basis of Close, Open, BotHat, and TopHat.

- Erode is neighborhood maximum filter that replaces the pixel at the center of the kernel area with the smallest (darkest) grayscale value found there. It therefore tends to increase the size of dark regions and decrease the size of light regions, "eroding" them.
- Dilate Takes its maximum value.

Open and Close Filters

Open and Close, derived from Erode and Dilate. The following figure shows TopHat, BotHat, Open, and Close:

Appendix I - Page **225**
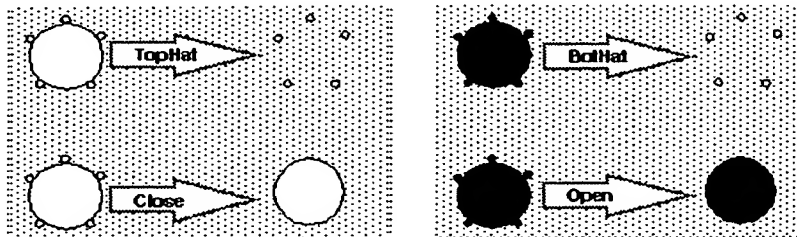
- Close dilates, then erodes—eliminating dark features smaller than the kernal. ?Intuition: assuming a light image on a dark background, tends to close up gaps. As a side effect, also increases the size
- Open erodes, then dilates—eliminating light features smaller than the kernel.

BotHat and TopHat Filters

BotHat and TopHat derive from open and close.

- BotHat calls Close to eliminate small, dark features, then removes all unchanged pixels from the resulting image. The final image contains?light? edges from the original image.
- TopHat calls Open to eliminate small, light features, then removes all unchanged pixels from the resulting image. The final image contains ?dark? edges from the original image.

Edge Direction and Edge Magnitude Filters

- EdgeDir performs ?Sobel Edge Detection? and produce an edge-magnitude image.
- EdgeMag XXX.

COGNEX In-Sight

◄ ▲ ► Image Processing: Warping an Image

In-Sight can warp an image and also translate a point from the warped image back to the unwarped image. To get started with warping, see *Warp Transforms: Getting Started*.

COGNEX In-Sight

◄ ▲ ► Image Processing: Reference

CompareImage	Trains and compares a stored buffer with an image region, saving a rectangular grayscale image segment and threshold template combination for later comparison with new image instances. The result of the comparison is typically zero pixel values where the match is within tolerance.
NeighborFilter	Processes a region with a filter that changes each pixel based on the values of neighboring pixels.
PointFilter	Processes a region with a filter that changes each pixel independently, ignoring adjacent pixels.
WarpImage	Maps a region of interest into a separate, rectangular image, warping the region into the rectangular result.

COGNEX In·Sight

◄ ▲ ► Image Processing: CompareImage function

Description   Trains and compares a stored buffer with an image region, saving a rectangular grayscale image segment and threshold template combination for later comparison with new image instances. The result of the comparison is typically zero pixel values where the match is within tolerance.

Heading   Vision Processing/Image

Syntax   CompareImage(Image, Fixture, Region, ..., Show)

Inputs	**Image**	Image data source, a reference to an Image structure. Default is A0.
	Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Region	Region of interest. **X** and **Y** define the top-left point by offsets from the
	—X	Fixture origin, in pixels. **High** and **Wide** are the height and width, in
	—Y	pixels. **Angle** is a rotation from the Fixture X axis, ± 360° CCW. See
	—High	*Defining a Region of Interest*.
	—Wide	
	—Angle	
	Normalization	Method for normalizing the grayscale values, one of:

- No Normalization
- Greylevel Scaling
- Histogram Fitting
- Mean/SDev fitting.

| | **Edge Tolerance** | Tolerance built into the template, in tenths of pixels, 0.10 through 10.00. |
| | **Exclude** | Tail to clip from histogram of input region. One of: |

Appendix I - Page **227**

Extremes

- No Tail Clipping
- Clip Dark Tail
- Clip Bright Tail
- Clip Both Tails.

Dark Tail	Percentage of dark histogram tail to ignore
Bright Tail	Percentage of bright histogram tail to ignore.
Train Flag	ON: train on OK or Run, even if setup unchanged; OFF: train on OK or Run only if setup changed.
Tolerance	Acceptable number of standard deviations, 0.00 through 10.00. Applied at run time.
Absolute Limit	Amount of pixel-level difference always accepted, 0.00 through 10.00. Applied at run time.
Image Shown	Flyover image to display. One of:

- Input Image
- Template Image
- Tolerance Image
- Output Image.

Show   Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Image**	On success, an Image structure, which stores the result image.
Emits	xxx	
Errors	Invalid parameter.	
Example	xxx	
Comments	None.	
See Also	Image structure, *Getting Started with Image Comparision*.	

COGNEX In·Sight

◄ ▲ ► Image Processing: NeighborFilter function

Description	Processes a region with a filter that changes each pixel based on the values of neighboring pixels.	
Heading	Vision Processing/Image	
Syntax	NeighborFilter(Image, Fixture, Region, Clear, Show, Script)	
Inputs	**Image**	Image data source, a reference to an *Image structure*. Default is A0.
	Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
	—**Row**	compensate for image-to-image variation in position, orientation, or both.
	—**Column**	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—**Theta**	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Region	Region of interest. **X** and **Y** define the top-left point by offsets from the
	—**X**	Fixture origin, in pixels. **High** and **Wide** are the height and width, in pixels.
	—**Y**	**Angle** is a rotation from the Fixture X axis, ± 360° CCW. See *Defining a*
	—**High**	*Region of Interest*.
	—**Wide**	
	—**Angle**	
	Operation	A filter. One of:

BotHat	Keeps dark features smaller than the kernel. Calls Close (to eliminate small, dark features) and then removes all unchanged pixels (leaving the dark features previously eliminated).
Close	Keeps bright features larger than the kernel. Calls Dilate (to "close" dark gaps in bright features) and then calls Erode (to eliminate the increase in feature size).
Dilate	Enlarges light features and reduces dark features. Sets the pixel at the center of the kernel to the maximum (brightest) grayscale value found there.
EdgeMagnitude	Applies an *edge-magnitude filter*, which returns an array of values representing the local edge strength, one value for each pixel in the ROI. Encodes the edge magnitudes between 0 and 63, where 0 is "no edge" and 63 is "strongest edge."
Erode	Enlarges dark features and reduces bright features. Sets the pixel at the center of the kernel to the minimum (darkest) grayscale value found there.
HiPass	Keeps features smaller than the kernel and destroys features larger than the kernel, attenuating low-frequency features. Employs a neighborhood-averaging difference filter that sets the pixel at the center of the kernel to the kernel mean (the LoPass result) minus the original value (to leave the differences).
LoPass	Keeps features larger than the kernel and destroys features smaller than the kernel, attenuating high-frequency features. Employs a neighborhood-averaging filter that sets the pixel at the center of the kernel to the mean within the kernel.

Appendix I - Page 229

	Open	Keeps dark features larger than the kernel. Calls Erode (to fuse bright gaps in dark features) and then calls Dilate (to eliminate the increase in feature size).
	TopHat	Keeps bright features smaller than the kernel. Calls Open (to eliminate small, bright features) and then removes all unchanged pixels (leaving the bright features previously eliminated).
Num Rows		Kernel height, 0 to 255, in pixels.
Num Cols		Kernel width, 0 to 255, in pixels.
Show		Selects the graphics to display, as described in _Flyover Graphics_:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Image**	On success, an _Image structure_, which stores the processed image.
Emits		The processed image.
Errors		Invalid parameter.
Comments		None.
See Also		Image structure, _Getting Started with Image Processing_

$$\rule{6cm}{1pt}$$ **COGNEX In-Sight**

◀ ▲ ▶ Image Processing: PointFilter function

Description		Processes a region with a filter that changes each pixel independently, ignoring adjacent pixels.
Heading		Vision Processing/Image
Syntax		PointFilter(Image, Fixture, Region, Clear, Show, Script)
Inputs	**Image**	Image data source, a reference to an _Image structure_. Default is A0.
	Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See _Working with Fixtures_.
	Region	Region of interest. **X** and **Y** define the top-left point by offsets from the
	—X	Fixture origin, in pixels. **High** and **Wide** are the height and width, in pixels.
	—Y	**Angle** is a rotation from the Fixture X axis, ± 360° CCW. See _Defining a_
	—High	_Region of Interest_.
	—Wide	
	—Angle	,
	Operation	A filter. One of:

Binarize ???
Clip   ???.

Equalize ???
Stretch ???

Minimum	Minimum gray value.
Maximum	Maximum gray value.
Threshold	Threshold grayscale value. −1 for automatic.
Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Image**	On success, an <u>Image structure</u>, which stores the processed image.
Emits	Nothing.	
Errors	Invalid parameter.	
Comments	None.	
See Also	<u>Image structure</u>, *Getting Started with Image Processing*	

=== **COGNEX In-Sight**

◄ ▲ ► Image Processing: WarpImage function

Description	Maps a region of interest into a separate, rectangular image, warping the region into the rectangular result.	
Heading	Vision Processing/Image	
Syntax	WarpImage(Image, Fixture, Region)	
Inputs	**Image**	Image data source, a reference to an <u>Image structure</u>. Default is A0.
	Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Region	Region of interest. **X** and **Y** define the top-left point by offsets from the
	—X	Fixture origin, in pixels. **High** and **Wide** are the height and width, in pixels.
	—Y	**Angle** is a rotation from the Fixture X axis, ± 360° CCW. **Curve** is the
	—High	angle of curvature, ± 360° CCW. (For curved regions, **Wide** is the arc
	—Wide	distance at the center of the region.) See *Defining a Region of Interest*.
	—Angle	
	—Curve	
Stores	**Image**	On success, an <u>Image structure</u>, which stores the warped image.
Emits	???	
Errors	Invalid parameter.	
Comments	The warped image is the result of a polar transformation. Any vision tool can use the result image as an input image, permitting all tools to process curved parts of the original image.	
See Also	<u>TransWarpToPixel</u>, *Getting Started with Warp Transforms*.	

COGNEX In-Sight

◄ ▲ ► Input/Output: Overview

The Input/Output functions read and write to external devices. This section introduces the Input/Output functions, discussing the following topics:

- Getting Started with the Input/Output Functions
- Input/Output Function Reference.

Functions in this category support DeviceNet and the ability to save a file to the PC host over the serial port.

COGNEX In-Sight

◄ ▲ ► Input/Output: Getting Started

This topic describes how to TBD.

► **To Set Up the Discrete I/O**

1. Acquire an appropriate image. For this exercise:
   a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
   b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
   c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

   To acquire a new image, hold ☐ and click ✕.

2. tbd.

COGNEX In-Sight

◄ ▲ ► Input/Output: Function Reference

DeviceNetRead	Reads data from DeviceNet via a serial port. Returns a DNData structure.
DeviceNetWrite	Writes data to DeviceNet via a serial port.
DNData structure	Stores DeviceNet data. Created by the DeviceNetRead function.
Event function	Executes on a specified internal, external, or manual trigger, updating all dependent cells. Returns an Event structure.
Event structure	Stores an event. Created by the Event function.

Appendix I - Page **233**

GetDeviceNetData Extracts data items read from DeviceNet.
GetEventValue   Returns the parameter value passed to Event().
ReadDiscrete   Reads a range of input bits at a specified event.
ReadDiscreteSingle Reads a single input bit at a specified event.
ReadSerial   Reads the serial input at a specified event.
SetEvent   Signals a specified event on execution. The source of soft triggers for AcquireImage() or
                     Event().
WriteDiscrete   Writes a value to a specified range of output bits.
WriteDiscreteSingle Writes a 0 or 1 to a single output bit.
WriteImageSerial   Copies the current image to a buffer, then sends the buffered image to a serial port.
WriteSerial   Writes its parameters to the serial output.

COGNEX In-Sight

◄ ▲ ► Input/Output: DeviceNetRead()

Description	Reads data from DeviceNet via a serial port. Returns a DNData structure.	
Heading	Input Output/Serial	
Syntax	DeviceNetRead(Event, Port, MapSpec)	
Inputs	**Event**	An event that triggers DeviceNetRead(). Normally a reference to an Event associated with the serial port. Rarely a reference to a cell containing AcquireImage or Button.
	Port	A serial port number, 0 or 1.
	MapSpec	A list of data specifiers delimited by colon characters (:). Each specifier has two parts: the starting byte (a zero-based byte offset) and a data type code. The single-character codes are:

    i   8- or 16-bit integer
    u 8- or 16-bit unsigned integer
    f   32-bit floating-point
    s   string

For example, "0i:2f:6u:7s" specifies that the first data item (at offset 0) is a 16-bit integer; the second item (at offset 2) is a floating-point value; the third (at offset 6) is a 1-byte unsigned integer; and the rest of the packet (starting at offset 7) is a string.

Stores	**DNData**	On success, a DNData structure, which stores the data from DeviceNet.
Emits	Nothing.	
Errors		

- Event is not a serial read event.
- Improperly formatted MapSpec parameter.
- Mismatch between the MapSpec and the incomming data, specified in the Serial I/O settings, through the DeviceNet Details option.

| Comments | The DeviceNet serial must be connected to a DIP DeviceNet serial communications gateway (CDN366). The DIP gateway and DeviceNet Details settings must match. That is, both must have the same packet-size setting. In addition, the serial communications settings must be the same on both sides. Hardware handshaking is required. |
| See Also | DeviceNetWrite, DNData structure, GetDeviceNetData. |

COGNEX In-Sight

◄ ▲ ► Input/Output: DeviceNetWrite()

Description	Writes data to DeviceNet via a serial port.
Heading	Input Output/Serial
Syntax	DeviceNetWrite(Port, MapSpec, Val ...)
Inputs	**Port**	A serial port number, 0 or 1.
	MapSpec	A list of data specifiers delimited by colon characters (:). Each specifier has two parts: the starting byte (a zero-based byte offset) and a data type code. The single-character codes are:

i   8- or 16-bit integer
u   8- or 16-bit unsigned integer
f   32-bit floating-point
s   string

For example, "0i:2f:6u:7s" specifies that the first data item (at offset 0) is a 16-bit integer; the second item (at offset 2) is a floating-point value; the third (at offset 6) is a 1-byte unsigned integer; and the rest of the packet (starting at offset 7) is a string.

	Val ...	A variable number of values to write to DeviceNet.
Returns	??.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	DeviceNetRead, DNData structure, GetDeviceNetData.

COGNEX In-Sight

◄ ▲ ► Input/Output: DNData structure

Description	Data read from DeviceNet by DeviceNetRead().
Values	**Value**	**Description**	**Access function**
	Data	An indexed list of data items.	GetDeviceNetData
Comments	None.
See Also	DeviceNetRead, DeviceNetWrite, DNData structure, GetDeviceNetData.

COGNEX In-Sight

◄ ▲ ► Input/Output: Event()

Description	Executes on a specified internal, external, or manual trigger, updating all dependent cells. Returns an Event structure.
Heading	Input Output
Syntax	Event(Trigger, Manual)
Inputs	**Trigger** An internally or externally generated event that updates dependent cells in the worksheet. To define a soft trigger, see SetEvent.
	Manual Enables the manual trigger (Control Pad☐ +✕). When enabled, the manual trigger does not preempt the specified Trigger. Event() consequently runs at either the manual trigger or the specified Trigger.
Returns	??.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	AcquireImage, Event structure, GetEventValue, ReadDiscrete, ReadDiscreteSingle, SetEvent, WriteDiscrete, WriteDiscreteSingle

COGNEX In-Sight

◄ ▲ ► Input/Output: Event structure

Description	Stores an event, used to update the worksheet. Created by Event().
Values	Event has no internal values that are read individually by Data Access functions. It stores an event on behalf of other functions requiring one.
Comments	The Event function defines an event and stores it in an Event structure.
See Also	Event function, GetEventValue, SetEvent.

COGNEX In-Sight

◄ ▲ ► Input/Output: GetDeviceNetData()

Description	Extracts data items read over DeviceNet.
Heading	Input Output/Serial
Syntax	GetDeviceNetData(DN Data, Index)
Inputs	**DN Data** A reference to data read by DeviceNetRead.
	Index A zero-based index for the data item to extract.
Returns	The data extracted from DN Data.

Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	DeviceNetRead, DeviceNetWrite, DNData structure.

COGNEX In-Sight

◄ ▲ ► Input/Output: GetEventValue()

Description	Returns the parameter value passed to Event().
Heading	Input Output
Syntax	GetEventValue(Event)
Inputs	**Event** ???.
Returns	The value passed to Event().
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	AcquireImage, Event function, Event structure, SetEvent.

COGNEX In-Sight

◄ ▲ ► Input/Output: ReadDiscrete()

Description	Reads a range of input bits at a specified event.
Heading	Input Output/Discrete
Syntax	ReadDiscrete(Event, Bit Number, Number of Bits)
Inputs	**Event** Trigger source, a cell-reference. Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	Bit Number First bit number to read, 0 through 7.
	Num of Bits Number of bits to read, 1 through 7.
Returns	The value represented by the specified bits.
Emits	Nothing.
Errors	Invalid parameter.
Comments	• This function lets you read a contiguous block of input bits as a unit, returning the value collectively represented. For example, if the start bit is 0, the total number of bits is 4, and all values are 1's, then this function returns 15.
	• To configure discrete input, see *Setting up Discrete Input*.
See Also	ReadDiscrete, WriteDiscrete, WriteDiscrete, ReadSerial, WriteSerial

COGNEX In-Sight

Appendix I - Page **237**

◄ ▲ ► Input/Output: ReadDiscreteSingle()

Description	Reads a single input bit at a specified event.
Heading	Input Output/Discrete
Syntax	ReadDiscreteSingle(Event, Bit Number)
Inputs	**Event** Trigger source, a cell reference. Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	Bit Number Bit number to read, 0 through 7.
Returns	The current value of the specified bit, 0.00 or 1.00
Emits	Nothing.
Errors	Invalid parameter.
Comments	To configure discrete input, see _Setting up Discrete Input_.
See Also	ReadDiscreteSingle, ReadDiscrete, WriteDiscrete, ReadSerial, WriteSerial

COGNEX In-Sight

◄ ▲ ► Input/Output: ReadSerial()

Description	Reads the serial output at a specified Event.
Heading	Input Output/Serial
Syntax	ReadSerial(Event, Port)
Inputs	**Event** Trigger source, a cell reference. Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	Port A serial port number, 0 or 1.
Returns	??The current data, as a string.
Emits	Nothing.
Errors	Invalid parameter.
Comments	• Accepts a variable-length, comma-delimited list of inputs, which are concatenated in the order given.
	• To configure the serial port, see _Setting Up the Serial Port_.
See Also	ReadDiscrete, ReadDiscreteSingle, WriteDiscrete, WriteDiscreteSingle

COGNEX In-Sight

◄ ▲ ► Input/Output: SetEvent()

Description	Signals a specified event on execution.
Heading	Input Output
Syntax	SetEvent(Generated Event)
Inputs	**Generated** The event to signal.
	Event
Returns	??.
Emits	Nothing.
Errors	Invalid parameter.
Comments	The source of soft triggers for AcquireImage() or Event().
See Also	AcquireImage, Event function, Event structure, GetEventValue.

COGNEX In·Sight

◄ ▲ ► Input/Output: WriteDiscrete()

Description	Writes a value to a specified range of output bits.
Heading	Input Output/Discrete
Syntax	WriteDiscrete(Bit Number, Number of Bits, Bit Value)
Inputs	**Bit Number** First bit number to write, 0 through 7.
	Number of Number of bit numbers to write, 1 through 7.
	Bits
	Bit Value Value to write. The valid range depends on the number of bits written. ?? Better name = Write Val??
Returns	1.0 on success or 0.0 on failure.
Emits	Nothing.
Errors	Invalid parameter.
Comments	This function permits writing a contiguous block of bits as a unit. For example, if the start bit is 0, the total number of bits is 4, and the value written is 15, then all four bits are set to 1's.
	To configure discrete output, see *Setting up Discrete Output*.
See Also	ReadDiscrete, ReadDiscreteSingle, WriteDiscrete, ReadSerial, WriteSerial

COGNEX In·Sight

◄ ▲ ► Input/Output: WriteDiscreteSingle()

Description	Writes a 0 or 1 to a specified output bit.
Heading	Input Output/Discrete
Syntax	WriteDiscreteSingle(Bit Number, Bit Value)
Inputs	**Bit Number** Bit number to write, 0 through 7.
	Bit Value Value to write, 0 or 1. ?Better Name = Write Val?
Returns	1.0 on success or 0.0 on failure.
Emits	Nothing.
Errors	Invalid parameter.
Comments	To configure discrete output, see *Setting up Discrete Output*.
See Also	ReadDiscrete, ReadDiscreteSingle, WriteDiscrete, ReadSerial, WriteSerial

COGNEX In-Sight

◄ ▲ ► Input/Output: WriteImageSerial()

Description	Copies the current image to a buffer, then sends the buffered image to a serial Port
Heading	Input/Output
Syntax	WriteImageSerial(Image, Port, Name)
Inputs	**Image** The image to save. A reference to an Image structure.
	Port Serial port. 0 or 1.
	Name A name for the file. A text string.
Returns	??? ???.
Emits	Nothing.
Errors	Image buffer occupied because a previous WriteImageSerial() operation is not done moving the image over the serial port.
Comments	To avoid suspending image acquistion and processing while saving the image—a potentially slow operation—WriteImageSerial():

1. Copies the current image to a buffer
2. Spools the buffered data through the specified serial port.

Calling WriteImageSerial() a second time before the file copy operation has completed yields an error.

See Also	None.

COGNEX In-Sight

◄ ▲ ► I/O: WriteSerial()

Description	Writes its parameters to the serial output.
Heading	Input Output/Serial
Syntax	WriteSerial(Port, String)
Inputs	**Port** A serial port number, 0 or 1.
	String String to write to **Port**.
Returns	1.0 on success or 0.0 on failure.
Emits	Nothing.
Errors	??Invalid ???.
Comments	Accepts a variable-length, comma-delimited list of inputs, which are concatenated in the order given.
	To configure the serial port for output, use **Serial()**.
See Also	ReadDiscrete, ReadDiscreteSingle, WriteDiscrete, WriteDiscrete, ReadSerial.

COGNEX In-Sight

◄ ▲ ► Math: Overview

The Mathematics functions offer standard mathematical operations, arranged under the following headings:

Logic	Logical AND, OR, and NOT; bitwise AND, OR, XOR, and NOT; range checking. For all logical functions, 0.00 is FALSE and non-0.00 is TRUE.
Lookup	Finds values in lists. Includes If(), for conditional formulas; also functions that get the row and column index of a worksheet cell.
Math	Arithmetical functions. Computes absolute values, sums, modulus, random numbers, square roots, and so on.
Operators	Relational operators. Their precedence is the same as in ANSI C. (Logical and bitwise operators are implemented as functions under the Logic heading.)
Trigonometry	Functions involving angles. Offers sine, cosine, tangent, arc sine, arc cosine, and arc tangent; also offers pi and conversions between radians and degrees.
Statistics	Standard statistical functions: arithmetical mean, standard deviation, maximum, and minimum.

This section introduces the Mathematics functions, discussing the following topics:

- Getting Started with the Mathematics Functions
- Operators Reference
- Mathematics Function Reference.

To assure compatibility of all numeric values in the worksheet, In-Sight represents all numbers as floating-point values. For example, it represents the integer 3 as 3.00. Similarly, for logical functions, it represents TRUE as non-0.00 and FALSE as 0.00. Eliminating the need for explicit numeric data types simplifies the task of developing spreadsheets for vision applications.

NOTES
- Some values are intrinsically integer. Supplying a fractional value for an integer parameter results in rounding to the nearest whole integer. For example, 3.1 rounds to 3.000, but 3.7 rounds to 4.000.
- In-Sight's underlying precision (32-bit floating-point) exceeds its displayed precision (three decimal places). Computations on floating-point values reflect the internal precision.
- An empty cell referenced as a numeric or Boolean value is interpreted as 0.0. For example, if A1 is empty, then Abs(A1) yields 0.0.

COGNEX In-Sight

◄ ▲ ► Math: Getting Started

Applications commonly use the Mathematics functions in formulas that process feature values from vision tools. This exercise illustrates the use of the If function—which returns one value if its first parameter evaluates to TRUE (non-0.00) and another value if it evaluates to FALSE (0.00)—to test the contrast value from a histogram. In this exercise, If() prints PASS when the contrast is greater than or equal to 50.

Otherwise, it prints FAIL. The results look like this:

A3	=If(G2>=50, "PASS", "FAIL")							Formula
	A	B	C	D	E	F	G	
0	@Image							
1		Thresh	NDark	NBright	AveDark	AveBright	Contrast	
2	@Hist	88.000	979.000	9021.000	44.243	129.044	84.801	Tested value
3	PASS							

Conditional result

2	@Hist	136.000	6563.000	3437.000	132.873	137.001	4.127
3	FAIL						

▶ **To Use If() to Display a Conditional Message**

1. Acquire an appropriate image. During development, you typically place a part sample under the test stand and snap a picture of it. Alternatively, to adjust the camera height, aperture, and focus, use live mode.
2. Before you can test a value, you have to have a value to test. A convenient value for this exercise is the contrast result from a histogram. To obtain it:
    a. From cell **A2**, click ✕ to open the Formula Builder.
    b. From **Vision Processing**, open the **Histogram** heading, and then open the ExtractHistogram property sheet.
    c. Click ✕ on the Region heading, switching to the interactive Region cursor. Use the Control Pad to adjust the region to enclose a feature in the image.
    d. Click ✕ to accept the region, and then click **OK** to close the property sheet. ExtractHistogram() computes the histogram, returns a Hist structure, and emits the contrast and other values.
3. Move to cell **A3**, and then click ✕ to open the Formula Builder.
4. From **Mathematics**, open the **Lookup** heading, select If, and click ✕ In-Sight places If() in the formula bar, leaving the Formula Builder open. Most Mathematics functions don't have property sheets. You directly edit their parameters.
5. For the first parameter (the condition), construct the formula **G2>=50** as follows:
    a. Click ⊞ , the relative reference button.
    b. Move the marquee to cell **G2**, the histogram contrast. Click ✕ to accept the reference.
    c. From **Mathematics**, open the **Operators** heading, select >=, and click ✕
    d. From the keypad, enter 50.
    e. From the keypad, enter a comma (the delimiter between the first and second parameters).
6. For the second parameter, which is the result when TRUE (non-0.00), click ⊞ to open the Text Entry dialog. Edit the string "PASS", and click **OK** to accept it.
7. From the keypad, enter a comma as a delimiter between the parameters.
8. For the third parameter, the result when FALSE, similarly edit "FAIL" and click **OK**.
9. From the keypad, enter a close-parenthesis character to mark the end of If()'s last parameter.
10. Click **OK** to close the Formula Builder. In-Sight evaluates the formula and prints PASS or FAIL in cell **A3**, depending on the value of cell **G2**. Provided the region of interest for ExtractHistogram() contains an appropriate feature, you probably get a PASS result.
11. You now have a working formula containing a Mathematics function and a Mathematics operator. To

get a better sense of its behavior, try this:

a. Click and release △ to open the System menu. Select **Triggers** to open the Triggers menu, and then enable the **Continuous** option. In-Sight acquires images as quickly as possible, reevaluating the If() formula for each image.

b. Put a blank sheet of paper in front of the camera. The contrast should drop below 50. The formula prints FAIL until you remove the paper.

COGNEX In-Sight

◄ ▲ ► Math: Operators Reference

The Operators heading lists the relational operators. Their precedence is the same as that in the ANSI C language. Logical and bitwise operations are available as functions, not as operators.

NOTE The Formula Builder lists only operators that are valid in the current context. Invalid operators are unavailable.

Symbol	Name	Description
+	Addition	Entered from Formula Builder keypad, not from the Operators list. The Sum function is an alternative in some contexts.
/	Division	Entered from Formula Builder keypad, not from the Operators list. Modulus is available through the Mod function.
=	Equality	—
<>	Inequality	—
>	Greater than	—
<	Less than	—
>=	Greater than or equal to	—
<=	Less than or equal to	—
*	Multiplication	Entered from Formula Builder keypad, not from the Operators list.
–	Subtraction	Entered from Formula Builder keypad, not from the Operators list.
–	Unary minus	Entered from Formula Builder keypad, not from the Operators list.

COGNEX In-Sight

◄ ▲ ► Math: Function Reference

Abs	Returns the absolute value of a specified value.
ACos	Returns the arc cosine angle in degrees.
And	Returns the logical AND of a variable-length list of values.
ASin	Returns the arc sine angle in degrees.
ATan	Returns the arc tangent angle in degrees, ± 90.
ATan2	Returns the arc tangent angle, ± 180 degrees, of the vector (DY,DX).
BitAnd	Returns the bitwise AND from its parameters.

BitNot	Returns the bitwise inverse of its parameter.
BitOr	Returns the bitwise OR of its parameters.
BitXor	Returns the bitwise exclusive OR of its parameters.
Ceiling	Truncates a floating-point value to an integer. Truncates up, yielding the next-largerinteger.
Choose	Returns the value of the indexed parameter. Takes an integer index followed by a variable number of floating-point values (the array).
Column	Returns the column number of a worksheet cell. Column A is 0.0, B is 1.0, and so on.
Cos	Returns the cosine, given an angle in degrees.
CountError	Returns number of errors in one or more cells or cell ranges.
Degrees	Converts from radians to degrees.
Exp	Returns e, the base of the natural logarithms, raised to a specified power.
If	Tests its first parameter, returning one value if it is TRUE (non-0.00) and another value if it is FALSE (0.00).
InRange	Returns TRUE (1.00) if a specified value is within a specified range.
Int	Truncates a floating-point value to an integer. Truncates down, yielding the next-smaller integer.
Max	Returns the maximum value from a variable number of values.
MaxI	Returns the zero-based index of the maximum value from a variable number of values.
Mean	Returns the mean (average) from a variable number of values.
Min	Returns the minimum value from a variable number of values.
MinI	Returns the zero-based index of the minimum value from a variable number of values.
Mod	Returns the remainder of Val2 divided by Val1.
Not	Returns the logical inverse of its argument.
Or	Returns the logical OR of a variable-length list of arguments.
Pi	Returns the value of pi.
Power	Returns a specified value to a specified power.
Radians	Converts from degrees to radians.
Rand	Returns a random number between 0.00 and 1.00, updating it at a specified event.
Round	Rounds a floating-point value the nearest integer value.
Row	Returns the row number of a worksheet cell.
SDev	Returns the standard deviation of a variable number of values.
Sin	Returns the sine, given an angle in degrees.
Sqrt	Returns the square root of a specified value.
Sum	Returns the sum of a variable number of values.
Tan	Returns the tangent, given an angle in degrees.
Trunc	Truncates a floating-point value to an integer. Truncates toward zero, yielding the original value without the fraction.

COGNEX In-Sight

◄ ▲ ► Math: Abs function

Appendix I - Page **245**

Description	Returns the absolute value of a specified value.
Heading	Mathematics/Math
Syntax	Abs(Val)
Inputs	**Val** An expression that evaluates to a floating-point value.
Returns	Absolute value of **Val**, as a floating-point value.
Emits	Nothing.
Errors	Invalid parameter, for instance a reference to a string or an error.
Example	Abs(–10) returns 10.00.
Comments	None.
See Also	Exp, Int, Mod, Power, Rand, Sqrt, Sum.

COGNEX In-Sight

◄ ▲ ► Math: ACos function

Description	Returns the arc cosine angle in degrees.
Heading	Mathematics/Trigonometry
Syntax	ACos(Val)
Inputs	**Val** An expression that evaluates to a floating-point value. Valid range, –1.00 through +1.00.
Returns	Arc cosine angle in degrees, as a floating-point value.
Emits	Nothing.
Errors	• **Val** is out-of-range • **Val** is invalid. For example, it is a reference to a string or to an error.
Example	ACos(–1) returns 180.00.
Comments	None.
See Also	ASin, ATan, ATan2, Cos, Degrees, Pi, Radians, Sin, Tan.

COGNEX In-Sight

◄ ▲ ► Math: And function

Description	Returns the logical AND of a variable-length list of values.
Heading	Mathematics/Logic
Syntax	And(Val1, Val2 ...)
Inputs	**Val1 ...** A variable-length, comma-delimited list of values. Each value can be a cell reference, a cell range, or a floating-point value.
Returns	Logical AND of **Val1 ... ValN**: TRUE (1.00) if all parameters are TRUE (non-0.00), and FALSE (0.00) if any parameter is FALSE (0.00).
Emits	Nothing.
Errors	#ERR if all parameters are invalid.

Example   And(1,2,3) returns 1.00.

                   And(0,1) returns 0.00.

Comments
- And() returns a value if at least one parameter is a valid numeric value. It ignores individual parameters that refer to empty cells or error cells and processes the valid parameters. If all parameters are invalid, And() returns an error.
- All non-0.00 inputs are treated as 1.00.

See Also   BitAnd, BitNot, BitOr, BitXor, InRange, Not, Or.

━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX In-Sight**

◄ ▲ ► Math: ASin function

Description   Returns the arc sine angle in degrees.
Heading   Mathematics/Trigonometry
Syntax   ASin(Val)
Inputs   **Val**   An expression that evaluates to a floating-point value. Valid range, −1.00 through +1.00.
Returns   The arc sine angle in degrees, as a floating-point value.
Emits   Nothing.
Errors   **Val** out of range.
Example   ASin(1) returns 90.00.
Comments   None.
See Also   ACos, ATan, ATan2, Cos, Degrees, Pi, Radians, Sin, Tan.

━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX In-Sight**

◄ ▲ ► Math: ATan function

Description   Returns the arc tangent angle in degrees, ± 90.
Heading   Mathematics/Trigonometry
Syntax   ATan(Val)
Inputs   **Val**   An expression that evaluates to a floating-point value.
Returns   Arc tangent angle, ± 90 degrees, as a floating-point value.
Emits   Nothing.
Errors   ??None.
Example   ATan(1) returns 45.00.
Comments   None.
See Also   ACos, ASin, ATan2, Cos, Degrees, Pi, Radians, Sin, Tan.

━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX In-Sight**

◄ ▲ ► Math: ATan2 function

Description	Returns the arc tangent angle, ± 180 degrees, of the vector (DY,DX).
Heading	Mathematics/Trigonometry
Syntax	ATan2(Val1, Val2)
Inputs	**Val1** DY, an expression that evaluates to a floating-point value.
	Val2 DX, an expression that evaluates to a floating-point value.
Returns	Arc tangent angle, ± 180 degrees, as a floating-point value.
Emits	Nothing.
Errors	??Val1 and Val2 both equal 0.00.
Example	ATan2(1,1) returns 45.00.
Comments	None.
See Also	ACos, ASin, ATan, Cos, Degrees, Pi, Radians, Sin, Tan.

COGNEX In-Sight

◄ ▲ ► Math: BitAnd function

Description	Returns the bitwise AND of its parameters.
Heading	Mathematics/Logic
Syntax	BitAnd(Val1, Val2)
Inputs	**Val1** An expression that evaluates to a floating-point value.
	Val2 An expression that evaluates to a floating-point value.
Returns	Bitwise AND of **Val1 ... ValN**.
Emits	Nothing.
Errors	??Val1 or Val2 is out of range.
Example	BitAnd(3,10) returns 2.00.
Comments	Processes only the lower the 16 bits of Val1 and Val2. Ignores the upper 16 bits. Result is a 16-bit unsigned integer.
See Also	And, BitNot, BitOr, BitXor, InRange, Not, Or.

COGNEX In-Sight

◄ ▲ ► Math: BitNot function

Description	Returns the bitwise NOT of its parameter.
Heading	Mathematics/Logic
Syntax	BitNot(Val)
Inputs	**Val** An expression that evaluates to a floating-point value.
Returns	Bitwise NOT of **Val**.
Emits	Nothing.

Errors	??**Val** is out of range.
Example	BitNot(1) returns −2.00.
Comments	Processes only the lower 16 bits of **Val**. Ignores the upper 16 bits. The result is a 16-bit unsigned integer.
See Also	And, BitAnd, BitOr, BitXor, InRange, Or, Not.

COGNEX In-Sight

◄ ▲ ► Math: BitOr function

Description	Returns the bitwise OR of its parameters.
Heading	Mathematics/Logic
Syntax	BitOr(Val1, Val2)
Inputs	**Val1** An expression that evaluates to a floating-point value.
	Val2 An expression that evaluates to a floating-point value.
Returns	Bitwise OR of **Arg1** and **Arg2**.
Emits	Nothing.
Errors	??**Val1** or **Val2** is out of range.
Example	BitOr(3,10) returns 11.00.
Comments	Processes only the lower 16 bits of **Val1** and **Val2**. Ignores the upper 16 bits. The result is a 16-bit unsigned integer.
See Also	And, BitAnd, BitNot, BitXor, InRange, Not, Or.

COGNEX In-Sight

◄ ▲ ► Math: BitXor function

Description	Returns the bitwise exclusive OR of its parameters.
Heading	Mathematics/Logic
Syntax	BitXor(Arg1, Arg2)
Inputs	**Arg1** An expression that evaluates to a floating-point value.
	Arg2 An expression that evaluates to a floating-point value.
Returns	Bitwise exclusive OR of **Arg1** and **argN**.
Emits	Nothing.
Errors	??**Val1** or **Val2** is out of range.
Example	BitXor(6, 10) returns 12.00.
Comments	Processes only the lower 16 bits of **Val1** and **Val2**. Ignores the upper 16 bits. The result is a 16-bit unsigned integer.
See Also	And, BitAnd, BitNot, BitOr, InRange, Not, Or.

COGNEX In-Sight

◄ ▲ ► Math: Ceiling function

Description	Truncates a floating-point value to an integer.
Heading	Mathematics/Math
Syntax	Int(Val)
Inputs	**Val** An expression that evaluates to a floating-point value.
Returns	An integer truncated from **Val**.
Emits	Nothing.
Errors	??None.
Example	• Int(5.995) returns 5.00. • Int(1.6) returns 1.00 • Int(−1.2) returns −2.00

Comments
- Ceiling() truncates up to the next-larger integer. In contrast, Int() truncates down to the next-smaller integer; Trunc() truncates toward zero; and Round() returns the nearest integer:

Value	Ceiling()	Int()	Round()	Trunc()
1.2	2.00	1.00	1.00	1.00
1.2	−2.00	−2.00	−1.00	−1.00
3.6	4.00	3.00	4.00	3.00
−3.6	−4.00	−4.00	−4.00	−3.00

- In-Sight expresses integer values in floating-point format. For example, the integer value 5 is displayed as 5.00.

See Also   Abs, Exp, Int, Mod, Power, Rand, Round, Sqrt, Sum, Trunc.

COGNEX In-Sight

◄ ▲ ► Math: Choose function

Description	Returns the value of the indexed parameter. Takes an integer index followed by a variable number of values (the array).
Heading	Mathematics/Lookup
Syntax	Choose(Index, Val0, Val1, ...)
Inputs	**Index** Index into the list. Zero-based, so **Val0** is 0, **Val1** is 1, and so on. **Val0 ...** A variable-length, comma-delimited list of values. Each value can be a cell reference, a cell range, a string, or a floating-point value.
Returns	Value of the parameter at **Index**, as a floating-point value.
Emits	Nothing.

Errors	• **Index** is not a positive integer, or does not exist in the list. For example, Choose (100, 2, 3) returns #ERR.
	• All members of the list are invalid.
	• The selected list item is an error. For example, Choose(0, #ERR, 4) returns #ERR, reflecting the indexed item.
Example	Choose(2, 7, 6, 13, 2) returns 13.00.
Comments	• Choose() assigns indexes to any parameters that are empty cells or error cells and propagates their empty or error value if selected. For example, Choose(0, #ERR, 10) returns #ERR, and Choose(0, *empty*, 10) returns *empty*.
	• If a list item is a cell range, then Choose() expands the range into its elements and assigns an index to each element. For example, Choose(1, A1:A4, D10) is interpreted as Choose(1, A1, A2, A3, A4, D10) and returns the value of A2, selected by the zero-based index of 1.
See Also	Column, If, MaxI, MinI, Row.

COGNEX In·Sight

◄ ▲ ► Math: Column function

Description	Returns the column number of a worksheet cell.
Heading	Mathematics/Lookup
Syntax	Column(Cell)
Inputs	**Cell** A reference to a worksheet cell.
Returns	The column number of **Cell**. Each column has a zero-based index, so column A is 0.0, B is 1.0, and so on.
Emits	Nothing.
Errors	None.
Example	Column(B2) returns 1.00.
Comments	None.
See Also	Choose, If, MaxI, MinI, Row.

COGNEX In·Sight

◄ ▲ ► Math: Cos function

Description	Returns the cosine, given an angle in degrees.
Heading	Mathematics/Trigonometry
Syntax	Cos(Angle)
Inputs	**Angle** An angle, in degrees. An expression that evaluates to a floating-point value.
Returns	Cosine of **Angle**, in degrees.
Emits	Nothing.

Errors	??None.
Example	Cos(180) returns −1.
Comments	None.
See Also	ACos, ASin, ATan, ATan2, Degrees, Pi, Radians, Sin, Tan.

COGNEX In-Sight

◄ ▲ ► Math: CountError function

Description	Returns number of errors in one or more cells or cell ranges.
Heading	Mathematics/Lookup
Syntax	CountError(Cell-Ref, [Cell-Ref ...])
Inputs	**Cell-Ref** A cell or cell range.
Returns	The number of errors in the specified cells.
Emits	Nothing.
Errors	None.
Comments	None.
See Also	Choose, If.

COGNEX In-Sight

◄ ▲ ► Math: Degrees function

Description	Converts from radians to degrees.
Heading	Mathematics/Trigonometry
Syntax	Degrees(Radians)
Inputs	**Radians** An angle in radians, expressed as a floating-point value.
Returns	Angle in degrees.
Emits	Nothing.
Errors	??None.
Example	Degrees(3.14) returns 179.908.
Comments	None.
See Also	ACos, ASin, ATan, ATan2, Cos, Pi, Radians, Sin, Tan.

COGNEX In-Sight

◄ ▲ ► Math: Exp function

Description	Returns e, the base of the natural logorithms, raised to a specified power.
Heading	Mathematics/Math
Syntax	Exp(Val)
Inputs	**Val** The exponant. An expression that evaluates to a floating-point value.
Returns	e raised to the **Val** power.
Emits	Nothing.
Errors	??**Val** is out of range? WHAT IS THE RANGE??
Example	Exp(1) returns 2.718.
Comments	None.
See Also	Abs, Int, Mod, Power, Rand, Sqrt, Sum.

COGNEX In-Sight

◄ ▲ ► Math: If function

Description	Tests its first parameter, returning one value if it is TRUE (non-0.00) and another value if it is FALSE (0.00).
Heading	Mathematics/Lookup
Syntax	If(Cond, Val1, Val2)
Inputs	**Cond** Condition for If(), an expression that evaluates to a floating-point value. TRUE is non-0.00 and FALSE = 0.00.
	Val1 Result, when **Cond** is TRUE (non-0.00).
	Val2 Result, when **Cond** is FALSE (0.00).
Returns	**Val1** If **Cond** is TRUE (non-0.00).
	Val2 If **Cond** is FALSE (0.00).
Emits	Nothing.
Errors	??None.
Example	• If(1, 1, 2) returns 1.00.
	• If(0, "True", "False") returns "False".
	• If(5, 3, 7) returns 3.00.
	• If($G$2>=50, "PASS", "FAIL") returns PASS or FAIL, depending on the current value of cell $G$2.
Comments	None.
See Also	Choose, Column, MaxI, MinI, Row, *Using the Mathematics Functions*.

COGNEX In-Sight

◄ ▲ ► Math: InRange function

Description	Returns TRUE (1.00) if a specified value is within a specified range.
Heading	Mathematics/Logic
Syntax	InRange(Val, Min, Max)
Inputs	**Val**	Value to test. An expression that evaluates to a floating-point value.
	Min	Minimum—the bottom of the range. An expression that evaluates to a floating-point value.
	Max	Maximum—the top of the range. An expression that evaluates to a floating-point value.
Returns	1.00	**Val** is within the range (**Min <= Val <= Max**).
	0.00	**Val** is not within the range.
Emits	Nothing.
Errors	??None.
Example	InRange(3, 1, 9) returns 1.00.
Comments	None.
See Also	And, BitAnd, BitNot, BitOr, BitXor, Or, Not.

COGNEX In-Sight

◄ ▲ ► Math: Int function

Description	Truncates a floating-point value to an integer.
Heading	Mathematics/Math
Syntax	Int(Val)
Inputs	**Val**	An expression that evaluates to a floating-point value.
Returns	An integer truncated from **Val**.
Emits	Nothing.
Errors	??None.

Example
- Int(5.995) returns 5.00.
- Int(1.6) returns 1.00
- Int(−1.2) returns −2.00

Comments
- Int() truncates down to the next-larger integer. In contrast, Ceiling() truncates up to the next-larger integer; Trunc() truncates toward zero; and Round() returns the nearest integer:

Value	Ceiling()	Int()	Round()	Trunc()
1.2	2.00	1.00	1.00	1.00
1.2	−2.00	−2.00	−1.00	−1.00
3.6	4.00	3.00	4.00	3.00
−3.6	−4.00	−4.00	−4.00	−3.00

- In-Sight expresses integer values in floating-point format. For example, the integer value 5 is displayed as 5.00.
- Int() is similar to the C-language function called Floor(). Both are the "opposite" of Ceiling().

See Also   Abs, Ceiling, Exp, Mod, Power, Rand, Round, Sqrt, Sum, Trunc.

== **COGNEX** In-Sight

◄ ▲ ► Math: Max function

Description	Returns the maximum value from a variable number of values.
Heading	Mathematics/Statistics
Syntax	Max(Val1, Val2 ...)
Inputs	**Val1 ...** A variable-length, comma-delimited list of values. Each value can be a cell reference, a cell range, or a floating-point value.
Returns	The largest value from the list.
Emits	Nothing.
Errors	#ERR if all parameters are invalid.
Example	Max(1,2,3,4) returns 4.00.
Comments	Max() returns a value if at least one parameter is a valid numeric value. It ignores individual parameters that refer to empty cells or error cells and processes the valid parameters. For example, Max(1, #ERR, 2, *empty*, 3) returns 3, which is the maximum of 1, 2, and 3. If all parameters are invalid, Max() returns an error.
See Also	MaxI, Mean, Min, MinI, SDev.

== **COGNEX** In-Sight

◄ ▲ ► Math: MaxI function

Description	Returns the zero-based index of the maximum value from a variable number of values.
Heading	Mathematics/Lookup
Syntax	MaxI(Val1, Val2 ...)
Inputs	**Val1 ...** A variable-length, comma-delimited list of values. Each value can be a cell reference, a cell range, or a floating-point value.
Returns	Zero-based index of the largest value from the list. If there is more than one instance of the largest value, returns the index of the first instance.
Emits	Nothing.
Errors	#ERR if all parameters are invalid.
Example	MaxI(1,2,3,4) returns 3.00.
Comments	• The indexes are zero-based, so the index of **Val1** is 0.0, the index of **Val2** is 1.0, and so on. • MaxI() returns a value if at least one parameter is a valid numeric value. Individual parameters that refer to empty cells or error cells are assigned indexes, but MaxI() ignores them when seeking the maximum. For example, MaxI(1, #ERR, 2, *empty*, 3) returns 4.00, which is the index of 3, the largest valid value on the list. If all parameters are invalid, MaxI() returns an error.

See Also   Choose, Column, If, Max, Min, MinI, Row.

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX** In-Sight

◄ ▲ ► Math: Mean function

Description	Returns the mean (average) from a variable number of values.
Heading	Mathematics/Statistics
Syntax	Mean(Val1, Val2 ...)
Inputs	**Val1 ...** A variable-length, comma-delimited list of values. Each value can be a cell reference, a cell range, or a floating-point value.
Returns	Arithmetic mean of from the list.
Emits	Nothing.
Errors	#ERR if all parameters are invalid.
Example	Mean(5, 7, 9) returns 7.00.
Comments	Mean() returns a value if at least one parameter is a valid numeric value. It ignores individual parameters that refer to empty cells or error cells and processes the valid parameters. For example, Mean(1, #ERR, 2, *empty*, 3) returns 2, which is the average of 1, 2, and 3. If all parameters are invalid, Mean() returns an error.
See Also	Max, Min, SDev.

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX** In-Sight

◄ ▲ ► Math: Min function

Description	Returns the minimum value from a variable number of values.
Heading	Mathematics/Statistics
Syntax	Min(Val1, Val2 ...)
Inputs	**Val1 ...** A variable-length, comma-delimited list of values. Each value can be a cell reference, a cell range, or a floating-point value.
Returns	Smallest value from the list.
Emits	Nothing.
Errors	#ERR if all parameters are invalid.
Example	Min(1,2,3,4) returns 1.00.
Comments	Min() returns a value if at least one parameter is a valid numeric value. It ignores individual parameters that refer to empty cells or error cells and processes the valid parameters. For example, Min(1, #ERR, 2, empty, 3) returns 1, which is the minimum of 1, 2, and 3. If all parameters are invalid, Min() returns an error.
See Also	Max, MaxI, Mean, MinI, SDev.

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX** In-Sight

◀ ▲ ▶ Math: MinI function

Description	Returns the zero-based index of the minimum value from a variable number of values.
Heading	Mathematics/Lookup
Syntax	MinI(Val1, Val2 ...)
Inputs	**Val1 ...** A variable-length, comma-delimited list of values. Each value can be a cell reference, a cell range, or a floating-point value.
Returns	Zero-based index of the smallest value from the list. If there is more than one instance of the smallest value, returns the index of the first instance.
Emits	Nothing.
Errors	#ERR if all parameters are invalid.
Example	MinI(1,2,3,4) returns 0.00.
Comments	• The indexes are zero-based, so the index of Val1 is 0.0, the index of Val2 is 1.0, and so on.
	• MinI() returns a value if at least one parameter is a valid numeric value. Individual parameters that refer to empty cells or error cells are assigned indexes, but MinI() ignores them when seeking the minimum. For example, MinI(1, #ERR, 2, empty, 3) returns 0.0, which is the index of 1, the smallest valid value on the list. If all parameters are invalid, MinI() returns an error.
See Also	Choose, Column, If, Max, MaxI, Min, Row.

COGNEX In-Sight

◀ ▲ ▶ Math: Mod function

Description	Returns the remainder of Val1 divided by Val2.
Heading	Mathematics/Math
Syntax	Mod(Val1, Val2)
Inputs	**Val1** An expression that evaluates to a floating-point value.
	Val2 An expression that evaluates to a floating-point value.
Returns	Modulus (remainder) of **Val1** divided by **Val2**, as a floating-point value.
Emits	Nothing.
Errors	**Val2** equals 0.00.
Example	Mod(5,12) returns 2.00.
Comments	None.
See Also	Exp, Int, Power, Rand, Sqrt, Sum.

COGNEX In-Sight

◀ ▲ ▶ Math: Not function

Description	Returns the logical inverse of its argument.
Heading	Mathematics/Logic
Syntax	Not(Val)
Inputs	**Val** An expression that evaluates to a floating-point value.
Returns	Logical NOT of **Val**: TRUE (1.00) for FALSE (0.00), and FALSE (0.00) for TRUE (non-0.00).
Emits	Nothing.
Errors	#ERR if **Val** is invalid, for example a reference to a string.
Example	Not(1) returns 0.00
	Not(0) returns 1.00.
Comments	All non-0.00 inputs are treated as 1.00 and therefore return 0.00.
See Also	And, BitAnd, BitNot, BitOr, BitXor, InRange, Or.

COGNEX In-Sight

◄ ▲ ► Math: Or function

Description	Returns the logical OR of a variable-length list of values.
Heading	Mathematics/Logic
Syntax	Or(Val1, Val2 ...)
Inputs	**Val1 ...** A variable number of values. Each value can be a cell reference, a cell range, or a floating-point value.
Returns	Logical OR of **Val1** ... **ValN**. TRUE (1.00) if any parameter is TRUE (non-0.00), and FALSE (0.00) if all parameters are FALSE (0.00).
Emits	Nothing.
Errors	#ERR if all parameters are invalid.
Example	Or(0,1) returns 1.00.
	Or(0,0) returns 0.00.
Comments	• Or() returns a value if at least one parameter is a valid numeric value. It ignores individual parameters that refer to empty cells or error cells and processes the valid parameters. If all parameters are invalid, Or() returns an error.
	• All non-0.00 inputs are treated as 1.00.
See Also	And, BitAnd, BitNot, BitOr, BitXor, InRange, Not.

COGNEX In-Sight

◄ ▲ ► Math: Pi function

Description	Returns the value of pi.
Heading	Mathematics/Trigonometry
Syntax	Pi
Inputs	None.
Returns	3.14159... (the 32-bit, floating-point value of pi).
Emits	Nothing.
Errors	None.
Example	Pi returns 3.141159..., displayed as 3.141.
Comments	None.
See Also	ACos, ASin, ATan, ATan2, Cos, Degrees, Radians, Sin, Tan.

COGNEX In-Sight

◄ ▲ ► Math: Power function

Description	Returns a specified value to a specified power.
Heading	Mathematics/Math
Syntax	Power(Base, Exp)
Inputs	**Base** The base, an expression that evaluates to a floating-point value.
	Exp The exponent, an expression that evaluates to a floating-point value.
Returns	**Base** to the **Exp** power.
Emits	Nothing.
Errors	??**Base** or **Exp** is an invalid number.
Example	Power(5,5) returns 3,125.
Comments	None.
See Also	Abs, Exp, Int, Mod, Rand, Sqrt, Sum.

COGNEX In-Sight

◄ ▲ ► Math: Radians function

Description	Converts from degrees to radians.
Heading	Mathematics/Trigonometry
Syntax	Radians(Degrees)
Inputs	**Degrees** An angle in degrees, to be converted to radians, expressed as a floating-point value.
Returns	The radians equivalent of **Degrees**, expressed as a floating-point value.
Emits	Nothing.
Errors	??None.
Example	Radians(180) returns 3.141, the 32-bit floating-point value of pi.
Comments	None.
See Also	ACos, ASin, ATan, ATan2, Cos, Degrees, Pi, Sin, Tan.

COGNEX In-Sight

◄ ▲ ► Math: Round function

Description	Rounds a floating-point value to the nearest integer value.
Heading	Mathematics/Math
Syntax	Round(Val)
Inputs	**Val** An expression that evaluates to a floating-point value.
Returns	An integer rounded from **Val**.
Emits	Nothing.
Errors	??None.
Example	

- Round(1.4) returns 1.00.
- Round(1.5) returns 2.00
- Round(1.6) returns 2.00.

Comments

- Round() returns the nearest integer. In contrast, Ceiling() truncates up to the next-larger integer; Int() truncates down to the next-smaller integer; and Trunc() truncates toward zero:

Value	Ceiling()	Int()	Round()	Trunc()
1.2	2.00	1.00	1.00	1.00
1.2	−2.00	−2.00	−1.00	−1.00
3.6	4.00	3.00	4.00	3.00
−3.6	−4.00	−4.00	−4.00	−3.00

- In-Sight expresses integer values in floating-point format. For example, the integer value 5 is displayed as 5.00.

See Also   Abs, Ceiling, Exp, Int, Mod, Power, Rand, Sqrt, Sum, Trunc.

COGNEX In-Sight

◄ ▲ ► Math: Rand function

Description	Returns a random number between 0.00 and 1.00, updating it at a specified event.
Heading	Mathematics/Math
Syntax	Rand(Event)
Inputs	**Event** An event that forces an update. Must be a reference to a cell containing AcquireImage (the default), Button, or Event
Returns	A random number between 0.00 and 1.00.
Emits	Nothing.
Errors	**Event** does not refer to a cell containing one of the listed functions.
Example	None.

Comments   Rand() must reside by itself, in its own cell. Other functions can refer to this cell to obtain
                the random value.
See Also   Abs, Exp, Int, Mod, Power, Sqrt, Sum.

COGNEX In-Sight

◄ ▲ ► Math: Row function

Description   Returns the row number of a referenced cell.
Heading   Mathematics/Lookup
Syntax   Row(Cell)
Inputs   **Cell**   A reference to a worksheet cell.
Returns   The row number of **Cell**.
Emits   Nothing.
Errors   None.
Example   Row(B2) returns 2.00.
Comments   None.
See Also   Choose, Column, If, MaxI, MinI.

COGNEX In-Sight

◄ ▲ ► Math: SDev function

Description   Returns the standard deviation from a variable number of values.
Heading   Mathematics/Statistics
Syntax   SDev(Val1, Val2 ...)
Inputs   **Val1 ...**   A variable-length, comma-delimited list of values. Each value can be a cell
                                   reference, a cell range, or a floating-point value.
Returns   The standard deviation value.
Emits   Nothing.
Errors   #ERR if all parameters are invalid.
Example   SDev(0.5, 1, 2, 1, 0.5) returns 0.61237, displayed as 0.612.
Comments   SDev() returns a value if at least one parameter is a valid numeric value. It ignores
                individual parameters that refer to empty cells or error cells and processes the valid
                parameters. For example, SDev(1, #ERR, 2, empty, 3) returns 1.00, which is the
                standard deviation of 1, 2, and 3. If all parameters are invalid, SDev() returns an error.
See Also   Max, Mean, Min.

COGNEX In-Sight

◄ ▲ ► Math: Sin function

Appendix I - Page **261**

Description	Returns the sine, given an angle in degrees.
Heading	Mathematics/Trigonometry
Syntax	Sin(Angle)
Inputs	**Angle** An angle, in degrees. An expression that evaluates to a floating-point value.
Returns	The sine of **Angle**.
Emits	Nothing.
Errors	??None.
Example	Sin(90) returns 1.00.
Comments	None.
See Also	ACos, ASin, ATan, ATan2, Cos, Degrees, Pi, Radians, Tan.

COGNEX In-Sight

◄ ▲ ► Math: Sqrt function

Description	Returns the square root, given a value.
Heading	Mathematics/Math
Syntax	Sqrt(Val)
Inputs	**Val** An expression that evaluates to a floating-point value. Must be non-negative.
Returns	Square root of **Val**.
Emits	Nothing.
Errors	**Val** is negative.
Example	Sqrt(9) returns 3.00.
Comments	None.
See Also	Abs, Exp, Int, Mod, Power, Rand, Sum.

COGNEX In-Sight

◄ ▲ ► Math: Sum function

Description	Returns the sum of a variable number of values.
Heading	Mathematics/Math
Syntax	Sum(Val1, Val2 ...)
Inputs	**Val1 ...** A variable-length, comma-delimited list of values. Each value can be a cell reference, a cell range, or a floating-point value.
Returns	The sum of **Val1 ... ValN**.
Emits	Nothing.
Errors	• The sum exceeds the maximum representable value. ??WHAT IS THAT?? • All inputs are invalid.
Example	Sum(2, 2.1, 3) returns 7.10.

Comments	Sum() returns a value if at least one parameter is a valid numeric value. It ignores (interprets as zero) individual parameters that refer to empty cells or error cells and processes the valid parameters. For example, Sum(1, #ERR, 2, empty, 3) returns 6, which is the sum of 1, 2, and 3. If all parameters are invalid, Sum() treats each a zero and returns 0.0.
See Also	Abs, Exp, Int, Mod, Power, Rand, Sqrt.

COGNEX In-Sight

◄ ▲ ► Math: Tan function

Description	Returns the tangent, given an angle in degrees.
Heading	Mathematics/Trigonometry
Syntax	Tan(Angle)
Inputs	**Angle** An expression that evaluates to a floating-point value.
Returns	The tangent of **Angle**.
Emits	Nothing.
Errors	??None.
Example	Tan(45) returns 1.00.
Comments	None.
See Also	ACos, ASin, ATan, ATan2, Cos, Degrees, Pi, Radians, Sin.

COGNEX In-Sight

◄ ▲ ► Math: Trunc function

Description	Truncates a floating-point value to an integer.
Heading	Mathematics/Math
Syntax	Trunc(Val)
Inputs	**Val** An expression that evaluates to a floating-point value.
Returns	An intereger truncated from **Val**.
Emits	Nothing.
Errors	??None.
Example	• Trunc(5.995) returns 5.00 • Trunc(1.6) returns 1.00 • Trunc(−1.2) returns −1.00

Comments
- Trunc() truncates toward zero. In contrast, Ceiling() truncates up to the next-larger integer; Int() truncates down to the next-smaller integer; and Round() returns the nearest integer:

Value	Ceiling()	Int()	Round()	Trunc()
1.2	2.00	1.00	1.00	1.00
1.2	−2.00	−2.00	−1.00	−1.00
3.6	4.00	3.00	4.00	3.00
−3.6	−4.00	−4.00	−4.00	−3.00

- In-Sight expresses integer values in floating-point format. For example, the integer value 5 is displayed as 5.00.

See Also   Abs, Ceiling, Exp, Int, Mod, Power, Rand, Sqrt, Sum, Round.

COGNEX In-Sight

◄ ▲ ► PatFind: Overview

The PatFind™ functions perform pattern-matching searches after building a model. The search supports two kinds of models, edge-based and area-based. An edge model extracts geometric boundaries from an image region and uses them to find instances of the model. Edge models tolerate variations in rotation, scale, and image intensity; they also tolerate partial occlusions. An area model extracts a specified image area to use as a template. It then measures the similarity between the template and an image region through normalized correlation, evaluating grayscale differences on a pixel-by-pixel basis. An area model tolerates variation in rotation and scale. Both kinds of model can be masked to exclude "don't care" regions from consideration.

This section introduces In-Sight's PatFind functions, discussing the following topics:

- Getting Started with Pattern Matching
- Building a Model
  - o Choosing an Algorithm
  - o Masking the Model
  - o Setting the Offset Point
- Searching an Image
  - o Setting Acceptance and Confusion Thresholds
  - o Setting the Number of Candidates to Find
  - o Setting an Angle Range
  - o Setting a Scale Range
- Sorting the Results
- PatFind Functions Reference.

Vision applications use pattern matching for feature extraction, particularly for features that are stable in size and shape. The PatFind functions are under the Vision Processing heading.

COGNEX In-Sight

◄ ▲ ► PatFind: Getting Started

Pattern matching is a model-based search technique. It compares a *model*—an example of the desired feature—to regions of an image to find regions that fit the model. In this topic, you extract a model from the image. Along the way, you learn some important In-Sight techniques.

► **To Extract a Model of a Hole**

1. Acquire an appropriate image. For this exercise:
   a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
   b. Click and release Δ, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the

image.
c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

To acquire a new image at any time, hold □ and click ✕.

2. Move to cell **A1** and click ✕ to open the Formula Builder.
3. From the **Vision Processing** category, select **PatFind**, and then click ✕ to open the FindPatterns property sheet.
4. To define the region of interest, select the **Region** heading and click✕. In the resulting dialog box, click the Region button. In-Sight hides the worksheet and displays the Region cursor, which you interactively adjust with the Control Pad. Repeatedly clicking ○ cycles through the cursor modes. One mode sets the region's postion, another its size, and another its angle:

Position   **Size**   **Angle**

Col   Wide

Row   High   Degrees CCW
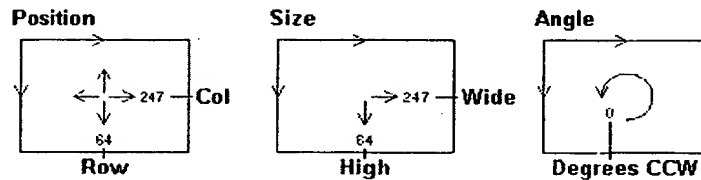
5. Using the Control Pad cursor, set the ROI position by dragging the Region cursor until the center of the region is at the center of the hole. Next, click ○ to change the Region cursor to its resize mode. Resize the ROI to surround the hole, clicking Up and Down to set the height and Right and Left to set the width. Exclude the background, like this:
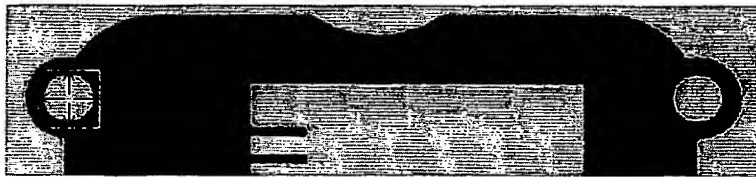
6. Click **OK**. FindPatterns() extracts a model from the region and stores it in a Patterns structure.

COGNEX In-Sight

◄ ▲ ► PatFind: Building a Model

In-Sight offers two kinds of models, built by FindPatterns. Parameters related to the model region also offer a mechanism for interactively defining the ROI. For example, to define a model, you define an ROI containing an example of it. To set up the ROI containing the model:

1. In the FindPatterns() dialog, select the Region heading and click ✕. In-Sight opens the Graphics Edit Options dialog:
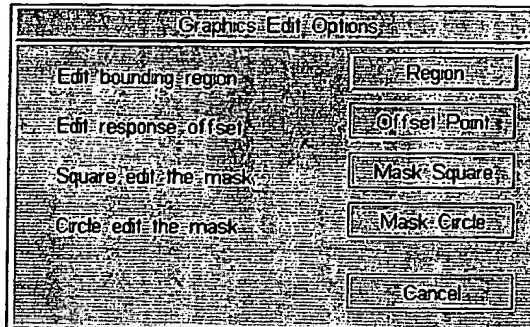
2. Select **Region** and click ✕. In-Sight hides the property sheet and replaces it with a rectangular cursor drawn on the image.
3. Using the cursor, move the ROI rectangle up and down and right and left until its top-left corner is at the top-left corner of the ROI.
4. To adjust the size, click the ◯, changing the cursor shape to a two-headed arrow, letting you resize the ROI horizontally or vertically. A model of a circular shape might look like this:
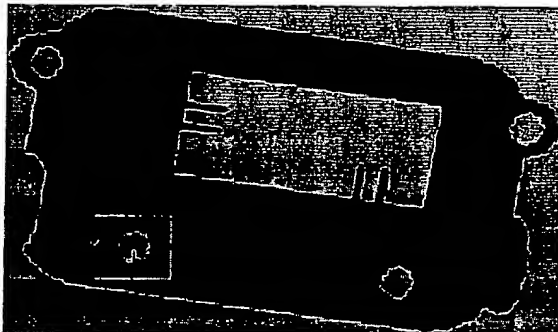
5. When satisfied with the ROI, click ✕ to accept it. In-Sight returns to the FindPatterns() property sheet.
    NOTE Clicking the △ button exits without changing the ROI.
6. Set other model-related parameters such as the offset point.
7. Click **Run** to ??build the model without closing the property sheet??; click **OK** to build the model and close the property sheet. In-Sight places a Patterns structure in the worksheet to store the model. Subsequent operations refer to this cell to obtain the model.

COGNEX In-Sight

◄ ▲ ► PatFind: Choosing the Algorithm

Discusses Non Linear. ??THIS TEXT GRABBED FROM ACUREAD. UNCERTAIN IF APPLICABLE. IS NON LIN THERE THE SAME AS NON LIN HERE??

A perfect OCR application would accurately read every sting every time. It would never reject even one character, no matter how severely degraded or occluded, as unreadable. It would never mistake one character for another.

A practical OCR application cannot attain perfect success and perfect accuracy at the same time. Real-world OCR is a trade-off between two conflicting goals:

- Robustness—the percentage of reads reported as successful, also called read rate. High robustness typically improves throughput and reduces operator intervention.
- Reliability—the percentage of reads that are actually correct, also called accuracy. High reliability typically improves yields and reduces costs.

Improving robustness by accepting low-scoring characters can harm reliability. Improving reliability by rejecting low-scoring characters can harm robustness. The right balance depends on your application. Poor reliability can be catastrophic, since misreads can cause production losses. Poor robustness is usually inconvenient but not catastrophic.

Most OCR methods lean one way or the other. A "restrictive" approach considers fewer character matches. Its caution makes it less likely to misread but more likely to fail a readable string. An "inclusive" approach considers more matches. Its aggressiveness makes it more likely to read a degraded string but more likely to misread. Figure 53 contrasts the two approaches.

Robustness and Reliability, in acuRead WILL ADAPT PICTURE

Reflecting this distinction, acuRead offers two reading methods, Standard OCR and Non Linear OCR™:

- Standard OCR reads the string once using a restrictive OCR technique. Any character that scores above the Standard acceptance threshold "passes." Virtual CheckSum is unavailable. Standard OCR is the method used in early versions of the Cognex acuWin and WinOCR software.
- Non Linear OCR reads the string twice, once with Standard OCR and again with Non Linear OCR, an inclusive technique. With Non Linear, acuRead (1) scores each character with both methods, (2) applies the Standard acceptance threshold to the Standard scores and Non Linear threshold to the Non Linear scores, (3) selects the Non Linear or Standard result as described in How acuRead Works on page 779. You'll probably find Non Linear OCR better than Standard OCR for reading severely degraded, partly occluded, or unevenly illuminated wafer scribes, especially if the string has a checksum.

The reading method determines the algorithm used to calculate the character and string scores. Non Linear OCR is more permissive than Standard OCR, often reporting a character value where Standard OCR would report a ? or * indicating "no match." Non Linear OCR also tends to report higher character and string scores. This fact—its higher scores—has no significance by itself, because the Standard and Non Linear scores reflect different quantities. But because the scores do differ, Standard and Non Linear OCR have distinct acceptance thresholds.

Non Linear OCR can often recognize characters that Standard OCR cannot read. It has more freedom to "guess"—and therefore more freedom to guess wrong. You can counterbalance this higher risk of false matches by selecting a judicious Non Linear acceptance threshold. In general, increasing an acceptance threshold increases your certainty that a character is correct when it passes, but also increases the chance of failing a readable character.

The best counterbalance is the use of a SEMI, BC412, or IBM412 checksum. If the string has a standard checksum, you can exploit the permissive aspect of Non Linear OCR while relying on the checksum to rule out false matches. Non Linear OCR is more likely than Standard OCR to return a complete string, but you can be confident of the result if the checksum passes. Combining Non Linear OCR with a standard checksum represents the best blend of robustness (read rate) versus reliability (accuracy).

Lacking a checksum, you can still use Non Linear OCR, but you should specify high values for both acceptance thresholds to screen out high-scoring but incorrect characters. A better approach for reading scribes that lack a checksum is to use Virtual Checksum, as described in Enabling a Standard or Virtual Checksum on page 791. Figure 54 shows the benefits of the various approaches.

Non Linear OCR with checksums

Non Linear OCR is slightly slower than Standard OCR due to the second read operation. To keep this cost small, acuRead does not duplicate the string and character finding overhead. Many applications can ignore the small difference in speed in exchange for the benefit of using Non Linear OCR and Virtual Checksum. Non Linear OCR incorporates Standard OCR without replacing it, so you can view Non Linear OCR as an enhanced version of Standard OCR.

COGNEX In·Sight

◄ ▲ ► PatFind: Masking the Model

??You can interactively define "don't care" regions though masking. This topic tells you how.

To edit the , click XXXX, yielding the following dialog:

Graphics Edit Options	
Edit bounding region	Region
Edit response offset	Offset Point
Square edit the mask	Mask Square
Circle edit the mask	Mask Circle
	Cancel
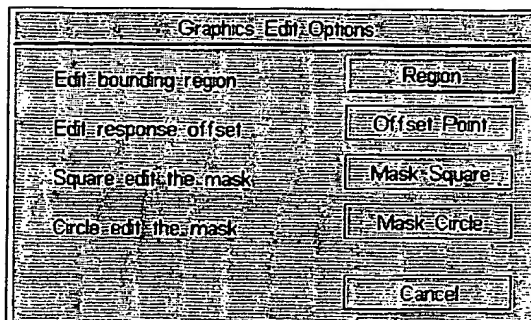
COGNEX In-Sight

◄ ▲ ► PatFind: Setting the Offset Point

??You can interactively set the offset point. This topic tells you how

To edit the offset point or mask, click XXXX, yielding the following dialog:

Graphics Edit Options	
Edit bounding region	Region
Edit response offset	Offset Point
Square edit the mask	Mask Square
Circle edit the mask	Mask Circle
	Cancel
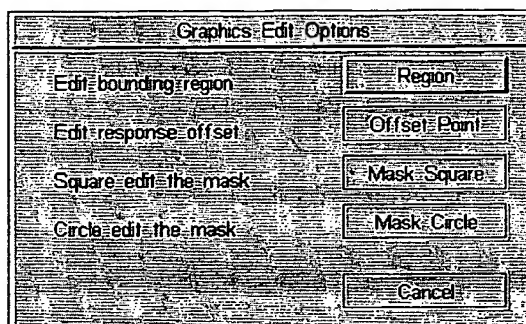
COGNEX In-Sight

◄ ▲ ► PatFind: Searching an Image

After building a model, you can search an image for it. Various parameters influence the search. Continuing the example, initiate the search as follows:

1. Open the FindPatterns property sheet.
2. Using the cursor, select **Model Data**. In-Sight defaults to interactive reference, hiding the property sheet.
3. ?? Move to a Patterns structure, then click ✕ to create the reference—which is, by default, absolute.
4. Click **Region** to define the ROI for the search. By default, In-Sight sets the ROI interactively. Use the Control Pad cursor, ✕ and △ buttons to surround the area of the image you wish to search:
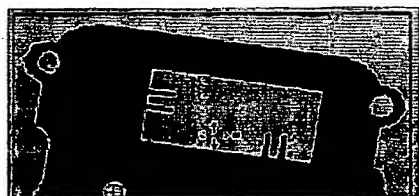
5. Edit the search parameters. For example, increase the Number to Find from 1 to 4.
6. Click **Run**. In-Sight searches the image, drawing a cross mark at the center of each instance found:
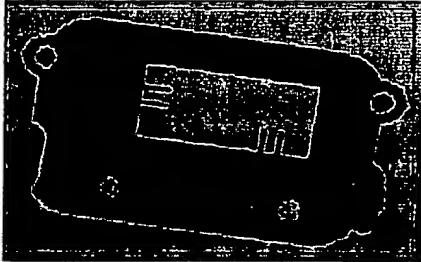
7.
8. Click **OK**, closing the property sheet. FindPatterns() emits result cells, from which other formulas can obtain specific result values:

Objects storing results

Image						
Pattern	Index	Row	Col	Angle	Scale	Score
Poses	0.000	326.034	200.962	0.000	100.000	100.000
	1.000	357.423	420.219	0.000	100.000	94.229
	2.000	#ERR	#ERR	#ERR	#ERR	0.000
	3.000	#ERR	#ERR	#ERR	#ERR	0.000

**Emitted results
for matched instances**

**#ERR results
(N2Find > actual number)**

FindPatterns() also generates a Patterns structure containing the data about the matched instances.

COGNEX In-Sight

◄ ▲ ► PatFind: Accept and Confus Thresholds

?? topic to discuss acceptance thresholds
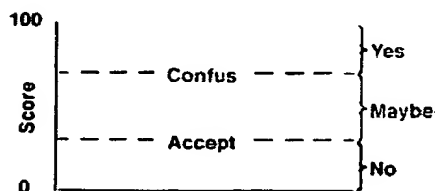
RAW TEXT, as a starting point

AcuSymbol searches for the symbol in the image to find the area of the image most similar to the symbol model. To measure the similarity, it calculates a correlation score between the image area and the model. A score of 0 indicates "no correlation," and a score of 100 indicates "perfect correlation." Increasing similarity between the model and area yields a higher score.

The acceptance and confusion thresholds are score values that control whether an image area matches or fails to match the model. The acceptance threshold defines the lowest score for a valid match. Image areas with scores below the acceptance value are not matches, so AcuSymbol immediately rejects them as potential symbol locations. The confusion threshold similarly defines the highest score for an invalid match. Areas with scores above the confusion value are matches, and AcuSymbol immediately accepts them as potential symbol locations. As shown in Figure 62, scores between the two thresholds might or might not match the model.

The search stops when the number of matches reaches N2Eval. If no match exceeds the acceptance threshold, then AcuSymbol zeros all Pose result values.

The valid range for both thresholds is 20 to 99, where 20 is a permissive standard and 99 is a strict standard. The default confusion value is 70, and the default acceptance value is 30. The defaults are appropriate for most applications and rarely require adjustment. If you do change the values, always set the confusion value above the acceptance value.

The major basis of choice for both thresholds is image quality, symbol quality, and the amount of separation between the symbol and its background. If your images contain speckle or other noise that might be mistaken for a symbol during the search phase, then increasing the thresholds can improve robustness (the percentage of decodes reported as successful) by decreasing the risk of false matches (which would all fail in the subsequent decode phase). But setting the thresholds too high can also harm robustness because it increases the risk of failing to find a decodable symbol. Low thresholds are often safe because the ECC checks during decoding are very reliable— a decode either works or fails outright, regardless of image or symbol quality.

For example, consider the pathological situation where a symbol is printed on a checkered background. Here, any random image area will exhibit some correlation to the model. If the thresholds are too low, then the search cannot distinguish the symbol from its background. If they are too high—forcing matches to have near-perfect correlations—then the search can miss genuine symbols due to minor miscorrelations.

COGNEX In·Sight

◄ ▲ ► PatFind: Setting the Number to Find

?? topic to discuss n2find

During the search phase, ?? keeps track of the number of matches greater than or equal to the confusion threshold. Generating a large pool of candidates increases the likelihood of finding a good candidate, but

generating an excessive number of candidates wastes time. You can limit the potential duration of the search by setting a maximum number of candidate responses (N2Eval).

A search operation stops when it the number of candidates reaches N2Eval. To reach N2Eval, all candidates must be less than the confusion threshold, so the find operation selects the best candidate between the acceptance and confusion thresholds. If none, it returns 0 (failure) and sets all Pose values to 0.

The default N2Eval value is suitable for many applications, but reducing N2Eval can slightly reduce search time.

COGNEX In-Sight

◄ ▲ ► PatFind: Setting an Angle Range

How to define an angle range **RAW TEXT, as a starting point**

To compensate for differences in orientation between the model and the image, ?? can apply an angle to the model before applying it to the image.

Model   Rotated Model
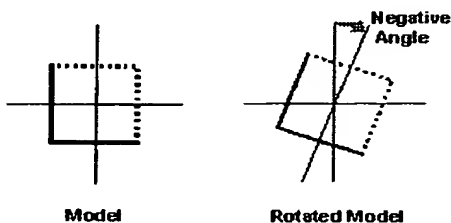
The angular rotation can any positive or negative value up to 180 degrees—sufficient to completely rotate the model with respect to the image.

AcuSymbol applies the angle and subsequently uses the angled model, so this operation has a minor, one-time computational cost. Nonetheless, you should if possible build a model that agrees with the expected image orientation, eliminating the need to rotate the model.

?? can also sweep the model through a range of angles:

Rotated Model   Rotated Model
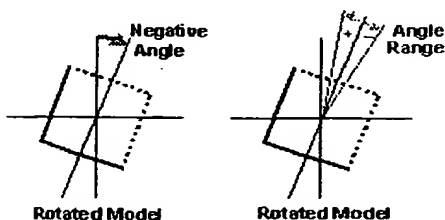
Starting at the current angle, it applies a specified rotational variance, in ± degrees, tolerated during a search. The valid range is 0 to 180 degrees—enough to sweep the model through all possible angles.

Searching an angle range can improve robustness if you cannot maintain a fixed symbol angle in the image. For example, consider a manufacturing process that exhibits some angular jitter. Here, specifying an angle range would let the decoder find the symbols despite the inconsistency. In general, increasing the range increases the search space, reducing speed. For best results, minimize rotational tolerance as much as possible.

TRANSLATE THIS TEXT TO FIXTURE A strategy for gaining the benefit of searching an angle range without excessive computational cost is to combine it with centering, which adjusts the angle to that of the last symbol successfully found. For details, see Centering the Symbol on page 932.

A side effect of applying an angle to the model is the need to increase the size of the region of interest. The exact increase depends on the angle, with the worst case governed by the diagonal:

Model rotated 0° (best case)   **Model rotated 45°** (worst case)

Minimum height and width = x   Minimum height and width = $x \cdot \sqrt{2}$
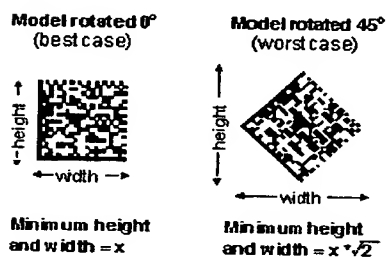
In general, the region of interest must contain the entire symbol—plus a three-module space on all sides for the quiet zone, plus an adequate positional tolerance, plus a margin sufficient for rotating or rescaling the model.

COGNEX In-Sight

◄ ▲ ► PatFind: Setting a Scale Range

How to define a scale range. **RAW TEXT, as a starting point.**

To compensate for differences in size between the model and the image, ??() can apply a scaling factor to the model before applying it to the image. The scale can vary from 80 to 150 percent, where the original model is 100 percent. This following figure shows the result of the size compensation.

Scale = 80%   **Model = 100%**   **Scale = 120%**
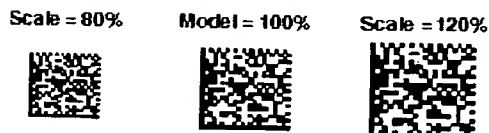
AcuSymbol applies the scaling once and then uses the scaled model. Rescaling incurs a one-time cost, but building the model at the expected size eliminates the adjustment.

As with the angle range operation, AcuSymbol can sweep the model through a scale range. It starts at the current scale and applies a specified variance, in ± percent, tolerated during a search. The maximum tolerance is 20%.
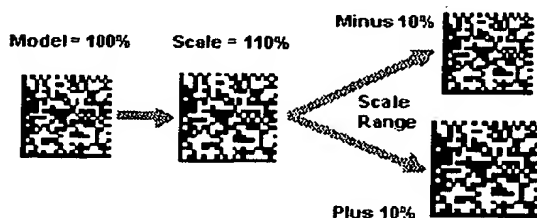
Searching a scale range can improve robustness if the symbol image varies in size, since the decoder can find the symbols despite the inconsistency. Increasing the range increases the search space, reducing speed, accuracy, and reliability. For best results, minimize the scale tolerance as much as possible.

A side effect that applies to the Angle Range also applies to scaling. The minimum size of the region of interest depends on the effective size of the model. When setting the region, include a margin adequate for the upper end of the scale range— including about three modules on all sides for the quiet zone.
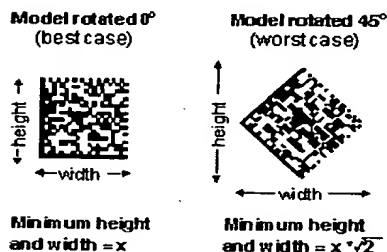
COGNEX In-Sight

◄ ▲ ► PatFind: Sorting Results

Discusses Sorting. D'oh!

COGNEX In-Sight

◄ ▲ ► PatFind: Functions Reference

FindPatterns	Extracts an area or edge model from a region; or searches for instances of a specified model; or extracts and searches in one step.
Patterns structure	Stores an area or edge model and the results of a pattern-match search. Each matched instance found has a zero-based index and a set of measured values Coordinates in Image units (pixels) unless converted by TransPatternsToWorld().
SortPatterns	Sorts a Patterns structure by specified criteria. Stores a new, sorted Patterns structure.

TransPatternsToWorld, under Coordinate Transforms, converts a Patterns structure to World coordinates. Results functions, under Data Access, that get values from a Patterns structure are GetAngle, GetCol, GetNFound, GetRow, GetScale, and GetScore.

COGNEX In-Sight

◄ ▲ ► PatFind: FindPatterns function

Description	Extracts an area or edge model from a region; or searches for instances of a specified model; or extracts and searchs in one step. Reports matches sorted by score. Stores the matched instances in a Patterns structure.
Heading	Vision Processing/PatFind
Syntax	FindPatterns(Image, Patterns, Fixture, Model Region, ... Show)
Inputs	**Image** — Image data source, a reference to an Image structure. Default is A0.
	Patterns — To build a new model, set to 0 (the default). To use an existing model, set to reference to a Patterns structure.
	Fixture — Fixture origin, defined by row, column, and angle offsets, typically to
	—Row — compensate for image-to-image variation in position, orientation, or
	—Column — both. **Row** and **Column** are offsets, in pixels from the image origin.
	—Theta — **Theta** is a rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Model Region — Region used as a model. **X** and **Y** define the top-left point by offsets
	—X — from the Fixture origin, in pixels. **High** and **Wide** are the height and
	—Y — width, in pixels. **Angle** is a rotation from the Fixture X axis,
	—High — ± 360° CCW. See *Defining a Region of Interest*.
	—Wide	
	—Angle	
	Offset — Reference point, an offset from the model center.
	Mode — Template matching algorithm to use. One of area model, edge model, or edge max.
	Coarseness — Distance between edges in the model: fine, medium, or coarse.
	Accuracy — Tradeoff between accuracy and speed: accurate, medium, or fast.
	ForceTrain — ON always trains; OFF trains only if the model has changed.
	Find Region — Region searched for the model. **X** and **Y** define the top-left point by

—X	offsets from the Fixture origin, in pixels. **High** and **Wide** are the height
—Y	and width, in pixels. **Angle** is a rotation from the Fixture X axis,
—High	± 360° CCW. See _Defining a Region of Interest_.
—Wide	
—Angle	
Number to Find	Number of candidates sought. See Setting the Number to Find.
Angle Range	Angle range to search. Valid range, ± 360° CCW from the model. See Setting an Angle Range. ·
Scale Tolerance	ON enables a scale tolerance during a search. See _Setting a Scale Range_
Accept Thresh	Acceptance threshold, the lowest acceptable score, 1 to 100. See _Setting Accept and Confus Thresholds_.
Confus Thresh	Confusion threshold, the highest score for an invalid instance, 1 to 100.
Show	Selects the graphics to display, as described in _Flyover Graphics_:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Patterns**	On success, a Patterns structure, which stores the matched instances.
Emits	**Index**	Zero-based index of a matched instance.
	NFound	Number of matched instances. Read by GetNFound.
	Row	Center row of instance, ??in what coordinate system? Read by GetRow.
	Col	Center column of instance. Read by GetCol.
	Angle	Angle of instance, in counterclockwise degrees relative to the model. Read by GetAngle.
	Scale	Scale of the instance, as a percentage of model. Read by GetScale.
	Score	Score of the instance, Read by GetScore.
Errors	Invalid parameter.	
Comments	None.	
See Also	TransPatternsToWorld	

COGNEX In·Sight

◄ ▲ ► PatFind: Patterns structure

Description   Stores an area or edge model and the results of a pattern-match search. Each matched instance has a zero-based index and a set of measured values. Coordinates in Image units (pixels) unless converted by TransPatternsToWorld.

Values	Value	Description	Access function
	Index	Index identifiying a particular matched instance. Zero-based, so the first instance is indexed 0.00, the second is indexed 1.00, and so on.	—
	Angle	Angle of instance relative to the model. In CCW degrees from the Image Row axis unless converted to World coordinates by TransPatternsToWorld().	GetAngle
	Column	Center point, column value. In pixels unless converted by TransPatternsToWorld().	GetCol
	NFound	Number of instances stored in the Patterns structure. Not converted by TransPatternsToWorld() because it is not a coordinate.	GetNFound
	Row	Center point, row value. In image units unless converted by TransPatternsToWorld().	GetRow
	Scale	Scale, as a percentage of the model. Not converted by TransPatternsToWorld() because it is not a coordinate.	GetScale
	Score	XXXXx. 0.00 to 100.00. Not converted by TransPatternsToWorld() because it is not a coordinate.	GetScore
Comments	None.		
See Also	TransPatternsToWorld		

COGNEX In-Sight

◄ ▲ ► PatFind: SortPatterns function

Description	Sorts a Patterns structure by specified criteria. Stores a new, sorted Patterns structure.
Heading	Vision Processing/PatFind
Syntax	SortPatterns(Patterns, Sort By, Number to Sort, Reference, Show)
Inputs	**Patterns** Search results. A reference to a Patterns structure, from FindPatterns. ?? Why is this the value 1 in the interface??
	Sort By Criterion for sorting the FindPatterns() results. One of:

- score
- row
- column
- angle
- scale
- distance, from the reference point
- grid (rows, then cols, from the reference point.

Number to Sort   Number of search results to sort.

Reference   Reference point, Row and Col, ... effects from the ... ...

Reference **—Row** **—Column** **—Angle**	Reference point, Row and Column are offsets from the image origin, in pixels. **Angle** is a rotation from the image row axis, 0 through 360 degrees CCW.	
Show	Selects the graphics to display, as described in *Flyover Graphics*: • Hide all • Result graphics only • Input and result graphics • Show all, which displays input, result, and chart (if any) graphics.	

Stores	**Patterns**	On success, a <u>Patterns structure</u>, which stores the sorted pattern-match results.
Emits	**Index**	Zero-based index of a matched instance.
	NFound	Number of matched instances. Read by <u>GetNFound</u>.
	Row	Center row of instance, ??what coordinate system? Read by <u>GetRow</u>.
	Col	Center column of instance. Read by <u>GetCol</u>.
	Angle	Angle of instance, in counterclockwise degrees relative to the model. Read by <u>GetAngle</u>.
	Scale	Scale of the instance, as a percentage of model. Read by <u>GetScale</u>.
	Score	Score of the instance, Read by <u>GetScore</u>.
Errors	Invalid parameter.	
Comments	None.	
See Also	<u>Sorting Pattern Match Results</u>, <u>FindPatterns</u>, <u>TransPatternsToWorld</u>	

COGNEX In-Sight

◄ ▲ ► Structures: Overview

The Structures functions provide a compact way to store a circle, cross, fixture, line, point, or region. Each function creates an structure that stores a set of values. For example, the Circle() function creates a Circle structure that stores the circle's center row, center column, and radius values. All of the values that define the circle then reside in a single cell, instead of consuming one cell each, simplifying the worksheet.

This section introduces the Structures functions and structures, discussing the following topics:

- Getting Started with the Structures Functions
- Structures Functions Reference.

A useful technique based on the Structures functions is to define structures for "global" items. Changing a value in the structure changes it in all formulas that refer to it, improving clarity and convenience. For example, suppose you want to apply more than one feature extraction function in the same region of interest. Creating a Region structure through the Region function makes it easy to define the region parameters and share them in all applicable contexts.

NOTE To access the Structures functions from the Formula Builder, open the **Advanced** heading.

COGNEX In·Sight

◄ ▲ ► Structures: Getting Started

In this topic, you create a Region structure with the Region function. You then use it to control the region for two other functions. Finally, you use the Control Pad to redefine the region, showing how both referencing functions share the same region. The result looks like this:

The techniques shown here apply equally well to all Structures functions.

► **To Share a Region**

1. Acquire an appropriate image. For this exercise:
   a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
   b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
   c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

   To acquire a new image at any time, hold □ and click ✕.

2. Leaving cell **A1** empty, move to cell **A2**, click ✕to open the Formula Builder.
   NOTE Try to leave a blank row above functions that emit formulas. In-Sight prints descriptive labels above emitted formulas if it has room for the labels. If the row above is occupied, it emits only the numeric values, without labels.
3. Open the **Advanced** heading, select the **Region** function from **Structures**, and click✕.
4. On the Region() property sheet, interactively define the region parameters by clicking the **Region** heading. In-Sight hides the worksheet and displays the Region cursor, which you adjust with the Control Pad. Outline an area containing a hole or other feature, click✕ to accept, and then click **OK** to close the property sheet. The Region function returns a Region structure and emits the region parameters:

Region object stores region parameters

Flyover graphic shows region

Emitted region values

5. Extract a histogram from the region:
   a. From cell A4, click ✕to open the Formula Builder. From **Vision Processing**, open the **Histogram** heading, select the ExtractHistogram function, and click✕ to open its property sheet.
   b. Click and hold ✕ on ExtractHistogram()'s **Region** heading, yielding the Enter menu. Cursor Up to **Relative**, and then release ✕
   c. Move the marquee to **B2** (the Row value), click and hold ○, and click cursor Right to select the range **B2:BF** (you can ignore **BG**, the Curve value). Release○, and then click✕ to accept the references and return to the ExtractHistogram() property sheet.
      NOTE To create references to the region, you must refer to the emitted row, column, and radius values. You cannot refer directly to the Region structure.
   d. Click OK. ExtractHistogram() returns a Hist structure and emits several statistical values.
6. Extract a blob from the same region:
   a. From A6, open the Formula Builder. Under **Vision Processing**, open the **Blob** heading, select ExtractBlobs, and click✕ to open its property sheet.
   b. As before, click and hold ✕ on the **Region** heading, opening the Enter menu. Select **Relative** and release ✕.
   c. Move the marquee to **B2**, click and hold ○, and click cursor Right to select the range **B2:BF** (the emitted region values). Release ○, and then click ✕ to accept the references and return to

Appendix I - Page 281

the property sheet.

    d. Click **OK** to close the property sheet. ExtractBlobs() returns a Blobs structure and emits values describing a blob.

7. Change the region, to show how ExtractHistogram() and ExtractBlobs() share the same parameters:

    a. Move to cell **A2**, which stores the Region structure, and click ✕ to open the Region() property sheet.

    b. Click Region()'s **Region** heading. As before, In-Sight displays the Region cursor.

    c. Adjust the region with the Control Pad. Outline an area containing a different feature, click ✕ to accept, and then click **OK** to close the property sheet. ExtractHistogram() and ExtractBlobs() update their values to reflect the new region of interest:

Region object (stores region)

Hist object (stores histogram)

Blobs object (stores blob features)

COGNEX In-Sight

◄ ▲ ► Structures: Function Reference

Circle function	Creates a Circle structure.
Circle structure	Stores a fixtured circle.
Cross function	Creates Cross structure.
Cross structure	Stores a fixtured cross-mark.
Fixture function	Creates a Fixture structure.
Fixture structure	Stores fixture coordinates.
Line function	Creates a Line structure.
Line structure	Stores a fixtured line.
Point function	Creates a Point structure.
Point structure	Stores a fixtured point.
Region function	Creates a Region structure.
Region structure	Stores a fixtured region.

Data Access functions that get individual values from the Circle, Cross, Fixture, Line, Point, and Region structures are GetAngle, GetCol, GetCurve, GetHigh, GetRadius, GetRow, and GetWide.

COGNEX In-Sight

◄ ▲ ► Structures: Circle function

Description	Creates a <u>Circle structure</u>.	
Heading	Advanced/Structures	
Syntax	Circle(Fixture, Circle, Show)	
Inputs	**Fixture**	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See _Working with Fixtures_.
	Circle	A circle, defined by its center point and radius. In Pixel or World units.
	—Row	
	—Column	
	—Radius	
	Show	Selects the graphics to display, as described in _Flyover Graphics_:

  - Hide all.
  - Result graphics only. Does nothing for Circle(), which has no result graphics.
  - Input and result graphics.
  - Show all, which displays input, result, and chart (if any) graphics.

Stores	**Circle**	On success, a <u>Circle structure</u>, which stores the circle.
Emits	**Row**	Center row. Read by <u>GetRow</u>.
	Col	Center column. Read by <u>GetCol</u>.
	Radius	Radius. Read by <u>GetRadius</u>.
Errors	Invalid parameter.	
Comments	One use is to define a circle in one place for multiple functions that refer to the same circle. Another use is to create a fixtured circle for functions that have circle parameters but not fixture parameters (<u>PlotCircle</u>, for example).	
See Also	<u>Circle structure</u>, <u>Cross function</u>, <u>Fixture function</u>, <u>Line function</u>, <u>Point function</u>, <u>Region function</u>.	

COGNEX In-Sight

◄ ▲ ► Structures: Circle structure

Description	Stores a fixtured circle.		
Values	**Value**	**Description**	**Access function**
	Row	Center row coordinate. In Pixel or World units.	<u>GetRow</u>
	Column	Center column coordinate. In Pixel or World units.	<u>GetCol</u>
	Radius	Radius. In Pixel or World units.	<u>GetRadius</u>

Comments   Created automatically by functions that store circle parameters or manually by the Circle
                function. The stored values are in Pixel or World units, reflecting the function that created
                this structure.

                NOTE To create references to the circle, you must refer to the emitted row, column, and
                      radius values. You cannot refer directly to the Circle structure to obtain its stored
                      values.
See Also   Circle function.

COGNEX In-Sight

◄ ▲ ► Structures: Cross function

Description   Creates a Cross structure.
Heading   Advanced/Structures
Syntax   Cross(Fixture, Cross, Show)
Inputs   **Fixture**   Fixture origin, defined by row, column, and angle offsets, typically to
                 —Row   compensate for image-to-image variation in position, orientation, or both.
                 —Column   **Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
                 —Theta   rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image
                                  origin). See *Working with Fixtures*.
                 Cross   A cross mark. **X** and **Y** define the center point by offsets from the Fixture
                 —X   origin, in pixels. **Angle** is a rotation of the cross mark from the Fixture X
                 —Y   axis, ± 360° CCW. **High** and **Wide** are the height and width, in pixels.
                 —Angle
                 —High
                 —Wide
                 Show   Selects the graphics to display, as described in *Flyover Graphics*:

                                  • Hide all.
                                  • Result graphics only. Does nothing for Cross(), which has no result
                                    graphics.
                                  • Input and result graphics.
                                  • Show all, which displays input, result, and chart (if any) graphics.

Stores   **Cross**   On success, a Cross structure, which stores the cross mark.
Emits   **Row**   Center row. Read by GetRow.
                 Col   Center column. Read by GetCol.
                 Angle   Angle, in counterclockwise degrees. Read by GetAngle.
                 High   Height, in pixels. Read by GetHigh.
                 Wide   Width, in pixels. Read by GetWide.
Errors   Invalid parameter.
Comments   One use is to define a cross mark in one place for functions that refer to the same cross
                 mark. Another use is to create a fixtured cross mark for functions that have cross mark
                 parameters but not fixture parameters (PlotCross, for example).

See Also　　　　Cross structure, Circle function, Fixture function, Line function, Point function, Region function.

═══════════════════════════════════════ **COGNEX** In-Sight

◄ ▲ ► Structures: Cross structure

Description　　Stores a fixtured cross mark.

Values

Value	Description	Access function
Row	Row coordinate. In Pixel or World units.	GetRow
Column	Column coordinate. In Pixel or World units.	GetCol
Angle	Angle. In Pixel or World units.	GetAngle
High	Height. In Pixel or World units.	GetHigh
Wide	Width. In Pixel or World units.	GetWide

Comments　　Created automatically by functions that store a cross mark or manually by the Cross function. The stored values are in Pixel or World units, reflecting the function that created this structure.

　　　　　　　NOTE To create references to the cross mark, you must refer to the emitted values. You cannot refer directly to the Cross structure.

See Also　　　　Cross function.

═══════════════════════════════════════ **COGNEX** In-Sight

◄ ▲ ► Structures: Fixture function

Description　　Creates a Fixture structure.

Heading　　　　Advanced/Structures

Syntax　　　　Fixture(Fixture, Show)

Inputs

Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
—Row	compensate for image-to-image variation in position, orientation, or both.
—Column	Row and Column are offsets, in pixels from the image origin. Theta is a
—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See Working with Fixtures.
Show	Selects the graphics to display, as described in Flyover Graphics:

- Hide all.
- Result graphics only. Does nothing for Fixture(), which has no result graphics.
- Input and result graphics. For Fixture(), shows the input point.
- Show all, which displays input, result, and chart (if any) graphics.

Stores　　　**Fixture**　　On success, a Fixture structure, which stores the fixture.

Emits	Row	Row value of fixture origin.
	Col	Column value of fixture origin
	Theta	Angle of fixture, in counterclockwise degrees from the image column axis. ??Currently Angle in interface. Is this changable??
Errors	Invalid parameter.	
Comments	None.	
See Also	Circle function, Cross function, Fixture structure, Line function, Point function, Region function.	

COGNEX In Sight

◄ ▲ ► Structures: Fixture structure

Description   Stores fixture coordinates.

Values	Value	Description	Access function
	Row	Row coordinate. An offset, in pixels from the image origin.	GetRow
	Column	Column oordinate. An offset, in pixels from the image origin.	GetCol
	Angle	Angle (theta). A rotation from the image row axis, + or − 360 degrees counterclockwise.	GetAngle
Comments	Created by the Fixture function.		
See Also	Fixture function.		

COGNEX In Sight

◄ ▲ ► Structures: Line function

Description	Creates a Line structure.
Heading	Advanced/Structures
Syntax	Line(Fixture, Line, Show)
Inputs	**Fixture** — Fixture origin, defined by row, column, and angle offsets, typically to
	—Row — compensate for image-to-image variation in position, orientation, or both.
	—Column — **Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta — rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Line — A line. **Row 0** and **Column 0** are the first end point, and **Row 1** and
	—Row0 — **Column 1** are the second end point. In Pixel or World units.
	—Column0	
	—Row1	
	—Column1	
	Show — Selects the graphics to display, as described in *Flyover Graphics*:

• Hide all

- Result graphics only. Does nothing for Line(), which has no result graphics.
- Input and result graphics. For Line(), shows the input line.
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Line**	On success, a Line structure, which stores the line.
Emits	**Row0**	First end point, row coordinate. Read by GetRow.
	Col0	First end point, column coordinate. Read by GetCol.
	Row1	Second end point, row coordinate. Read by GetRow.
	Col1	Second end point, column coordinate. Read by GetCol.
Errors		Invalid parameter.
Comments		One use is to define a line in one place for multiple functions that refer to the same line. Another use is to create a fixtured line for functions that have line parameters but not fixture parameters (PlotLine, for example).
See Also		Circle function, Cross function, Fixture function, Line structure, Point function, Region function.

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX** In·Sight

◄ ▲ ► Structures: Line structure

Description	Stores a fixtured line.		
Values	**Value**	**Description**	**Access function**
	Row0	First end point, row coordinate. In Pixel or World units.	GetRow, index 0
	Col0	First end point, column coordinate. In Pixel or World units.	GetCol, index 0
	Row1	Second end point, row coordinate. In Pixel or World units.	GetRow, index 1
	Col1	Second end point, column coordinate. In Pixel or World units.	GetCol, index 1
Comments	Created automatically by functions that store a line or manually by the Line function. The stored values are in Pixel or World units, reflecting the function that created this structure.		
	NOTE To create references to the line, you must refer to the emitted row and column values. You cannot refer directly to the Line structure.		
See Also	Line function.		

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **COGNEX** In·Sight

◄ ▲ ► Structures: Point function

Description	Creates a Point structure.	
Heading	Advanced/Structures	
Syntax	Point(Fixture, Point, Show)	
Inputs	**Fixture**	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Point	A point. In Pixel or World units.
	—Row	
	—Column	
	Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all.
- Result graphics only. Does nothing for Point(), which has no result graphics.
- Input and result graphics. For Point(), shows the input point.
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Point**	On success, a Point structure which stores the point.
Emits	**Row**	X or Row value. Read by GetRow.
	Col	Y or Col value. Read by GetCol.
Errors	Invalid parameter.	
Comments	One use is to define a point in one place for multiple functions that refer to the same point. Another use is to create a fixtured point for functions that have point parameters but not fixture parameters (PlotPoint, for example).	
See Also	Circle function, Cross function, Fixture function, Line function, Point structure, Region function.	

COGNEX In-Sight

◄ ▲ ► Structures: Point structure

Description	Stores fixtured point.		
Values	**Value**	**Description**	**Access function**
	Row	Row coordinate. In Pixel or World units.	GetRow
	Column	Column coordinate. In Pixel or World units.	GetCol
Comments	Created automatically by functions that store a point or manually by the Point function. The stored values are in Pixel or World units, reflecting the function that created this structure.		

> NOTE To create references to the point, you must refer to the emitted row and column values. You cannot refer directly to the Point structure.

See Also	Point function.

COGNEX In-Sight

◄ ▲ ► Structures: Region function

Description		Creates a Region structure.
Heading		Advanced/Structures
Syntax		Region(Fixture, Region, Show)
Inputs	**Fixture**	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Region	A rectangular or curved region. **X** and **Y** define the top-left point by offsets
	—X	from the Fixture origin, in pixels. **High** and **Wide** are the height and width,
	—Y	in pixels. **Angle** is a rotation from the Fixture X axis, ± 360° CCW. **Curve**
	—High	is the angle of curvature, ± 360° CCW. (For curved regions, **Wide** is the
	—Wide	arc distance at the center of the region.) See *Defining a Region of*
	—Angle	*Interest*.
	—Curve	
	Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all.
- Result graphics only. Does nothing for Region(), which has no result graphics.
- Input and result graphics.
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Region**	On success, a Region structure, which stores the region coordinates.
Emits	**Row**	Row value of region origin. Read by GetRow.
	Col	Column value of region origin. Read by GetCol.
	High	Height, in pixels. Read by GetHigh.
	Wide	Width, in pixels. Read by GetWide.
	Angle	Angle, in counterclockwise degrees, of the region. Read by GetAngle.
	Curve	The angle of curvature, in counterclockwise degrees. Read by GetCurve.
Errors		Invalid parameter.
Comments		One use is to define a region in one place for multiple functions that refer to the same region. Another use is to create a fixtured region for functions that have region parameters but not fixture parameters (PlotRegion, for example).
See Also		Region structure, *Getting Started with the Structures Functions.*

COGNEX In-Sight

◄ ▲ ► Structures: Region structure

Description	Stores a fixtured region.		
Values	**Value**	**Description**	**Access function**
	Row	Row coordinate. An offset from the Fixture origin, in pixels.	GetRow
	Column	Column coordinate. An offset from the Fixture origin, in pixels.	GetCol
	Angle	Angle. A rotation from the Fixture X axis, + or − 360 degrees counterclockwise.	GetAngle
	High	Height. In pixels.	GetHigh
	Wide	Width. In pixels.	GetWide
	Curve	Angle of curvature, in counterclockwise degrees.	GetCurve
Comments	Created automatically by functions that store a region or manually by the Region function.		

NOTE To create references to the region, you must refer to the emitted row, col, high, wide, angle, and curve values. You cannot refer directly to the Region structure to obtain its stored values.

See Also   Region function, *Getting Started with the Structures Functions.*

COGNEX In-Sight

◄ ▲ ► System Functions: Introduction

The System functions let you high-level operations from within the worksheet. For example, you can save a job or load one. This section discusses the following topics:

- System Functions Reference

The Systems functions are under Advanced.

COGNEX In-Sight

◄ ▲ ► System Functions: Reference

EnterLive	Puts In-Sight in live mode or graphics mode.
LoadJob	Loads a specified job at a specified event.
SaveJob	Saves a specified job at a specified event.
SetOnline	Puts In-Sight in online mode or offline mode.

COGNEX In-Sight

◄ ▲ ► System Functions: EnterLive()

Description	Puts In-Sight in live mode.	
Heading	Advanced/System	
Syntax	EnterLive(Event)	
Inputs	**Event**	An event that triggers EnterLive(). Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
Returns	???	???.
Emits	Nothing.	
Errors	None.	
Comments	????	
See Also	None.	

COGNEX In-Sight

◄ ▲ ► System Functions: LoadJob()

Appendix I - Page **291**

Description	Loads a specified job at a specified event.	
Heading	Advanced/System	
Syntax	LoadJob(Event, JobID)	
Inputs	**Event**	An event triggers LoadJob(). Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	JobID	Job ID. 0 through 19.
Returns	???	???.
Emits	Nothing.	
Errors	None.	
Comments	????	
See Also	SaveJob	

COGNEX In-Sight

◄ ▲ ► System Functions: SaveJob()

Description	Saves a specified job at a specified event.	
Heading	Advanced/System.	
Syntax	SaveJob(Event, JobID, Name)	
Inputs	**Event**	An event that triggers SaveJob(). Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	JobID	Job ID. 0 through 19.
	Name	A name for the job. A text string up to 15 characters long.
Returns	???	???.
Emits	Nothing.	
Errors	None.	
Comments	????	
See Also	LoadJob	

COGNEX In-Sight

◄ ▲ ► System Functions: SetOnline()

Description	Puts In-Sight in online mode or offline mode.	
Heading	Advanced/System	
Syntax	SetOnline(Event, Mode)	
Inputs	**Event**	An event that triggers SaveJob(). Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	Mode	1 for online, 0 for offline.
Returns	???	???.
Emits	Nothing.	
Errors	None.	

1. Using the Control Pad cursor, move to any cell in the row or column to be selected.
2. Holding down the ○ button, click cursor Left to select a row or cursor Up to select a column. In-Sight highlights the selected row or column:

Selected row	Selected column
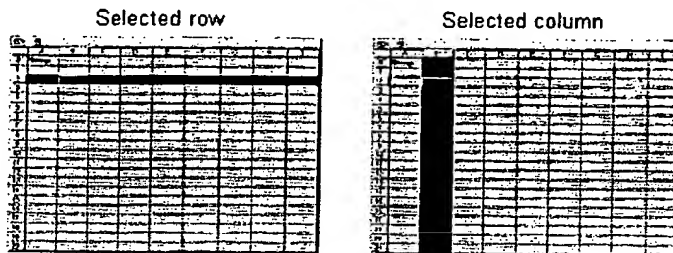
3. Release ○. In-Sight opens the Edit menu.
4. From the Edit menu, select a command and click ✕.

═══► **COGNEX** In-Sight

◄ ▲ ► Worksheet: Cut, Copy, and Paste Cells

Cutting, copying, and pasting are important operations for editing worksheets. They conform to the common conventions and use a buffer called a *clipboard* for temporary storage.

NOTE The difference between <u>absolute</u> and <u>relative</u> cell references is important when copying cells. When you copy a cell that contains a formula, In-Sight creates a new formula in the new cell. Absolute references remain unchanged, but relative references are edited to preserve equivalent offsets.

NOTE You can use the copy-and-paste sequence to move cells between jobs. To do so, copy from the source job, load the target job, and paste into it.

► **To Cut Cells**

To remove the contents of a cell without copying it to the clipboard, <u>clear it</u> instead of cutting it.

1. <u>Select the cell or range of cells</u> you wish to remove from the worksheet.
2. Click and release the ○ button, opening the <u>Edit menu</u>.
3. Select **Cut** and click ✕ In-Sight removes the selected cells from the worksheet and places in a clipboard, from which you can subsequently paste them.

► **To Copy Cells**

1. <u>Select the cell or range of cells</u> you wish to copy.
2. Click and release ○, opening the <u>Edit menu</u>.
3. Select **Copy** and click ✕ Leaving the selected cells in place, In-Sight copies them to a clipboard, from which you can subsequently paste them.

▶ **To Paste Cells**

NOTE Pasting into an occupied cell overwrites its contents without warning.

1. Cut or copy a cell or range of cells, as described above.
2. Using the Control Pad, navigate to an empty or occupied worksheet cell.
3. Click and release O, opening the Edit menu.
4. Select **Paste** and click ✕ In-Sight copies the cells from the clipboard to the worksheet, overwriting any existing data. Absolute references remain unchanged, but relative references are adjusted to maintain equivalent relative relationships.

COGNEX In-Sight

◄ ▲ ► Worksheet: Clearing Cells

Clearing a cell eliminates its contents without saving a copy to the clipboard. Cutting a cell, in contrast, keeps a copy, typically as a way to move the cell through a subsequent paste. Use clear instead of cut to conserve memory when you want to permanently remove cells.

▶ **To Clear Cells**

NOTE The deletion is permanent. You cannot undo it.

1. Select the cell or range of cells you wish to clear from the worksheet.
2. Click and release the O button, opening the Edit menu.
3. Select **Clear** and click ✕. In-Sight clears the cells.

COGNEX In-Sight

◄ ▲ ► Worksheet: Enable or Disable a Cell

Disabling a cell locks its value but leaves it in the worksheet. In an application, you typically disable a cell that implements a one-time set-up operation to prevent further evaluation. Calibration values, for example, are good candidates for being disabled. During development, you can disable a cell isolate it, for example to help debug the worksheet.

You can enable or disable a cell either manually or dynamically. You manually enable or disable through a toggle in a dialog. You dynamically enable or disable a cell through a relative or absolute reference to another cell, letting you lock or unlock the cell at run time. If the referenced cell evaluates to 0.0, then the referencing cell is disabled. If it evaluates to non-0.0, then the cell is enabled.

▶ **To Enable or Disable a Cell or Cell Range**

1. <u>Select</u> or cell range.
2. Click and release the ○ button, opening the <u>Edit menu</u>.
3. Select **Cell State**. In-Sight opens the Cell State dialog:

Current state ── Enabled

Manually enable or disable the cell ── ☐ Enable

Relative ── Dynamically enable by relative reference

Absolute ── Dynamically enable by absolute reference

Cell State
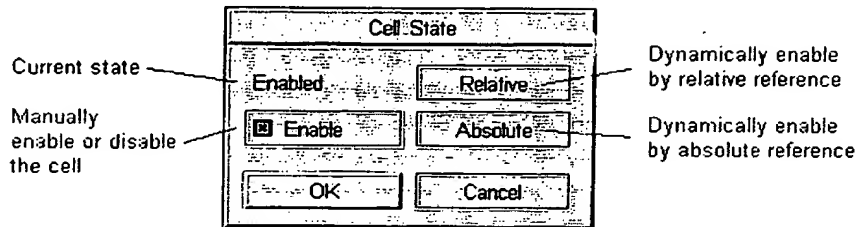
OK　　Cancel

4. Toggle **Enable** ON to enable the cell or OFF to **disable** it.
5. Click OK or Cancel.

COGNEX In-Sight

◄ ▲ ► Worksheet: Hide Rows or Columns

In-Sight lets you hide rows or columns in the worksheet. A common use is to simplify the appearance of the worksheet, either for an operator interface or as a convenience during development.

▶ **To Hide Rows**

1. <u>Select the cell or range of cells</u> to hide.
2. Click and release the ○ button, opening the <u>Edit menu</u>.
3. Select **Hide**. In-Sight hides the rows.

▶ **To Unhide Rows**

1. XXXX
2. XXXX.

▶ **To Hide Columns**

1. <u>Select the cell or range of cells</u> to hide.
2. Click and release ○, opening the <u>Edit menu</u>.
3. Select **Hide**. In-Sight hides the rows.

▶ **To Unhide Columns**

1. XXXX
2. XXXX.

COGNEX In-Sight

◄ ▲ ► Worksheet: Insert or Remove Rows

Inserting a row adds a new row ??below?? the current row. Removing a row permanently eliminates it.

NOTE These operations are available only for rows, not for columns.

► **To Insert a Row**

1. Using the Control Pad, select any cell in the row ??above?? the one you want to insert.
2. Click and release the O button, opening the Edit menu.
3. Select **Insert Rows**. In-Sight inserts one row ?below? the selected cell.

► **To Remove a Row**

CAUTION The deletion is permanent. You cannot undo it.

1. Using the Control Pad, select one or more cells in one ??or more?? rows.
2. Click and release O, opening the Edit menu.
3. Select **Remove Rows**. In-Sight removes the rows containing the selected cells.

COGNEX In-Sight

◄ ▲ ► Worksheet: Row Height, Column Width

You can set the row height or column width to a convenient value. ??Same for all, or row-by-row and col-by-col??

► **To Set the Row Height**

1. Click and release the O button, opening the Edit menu.
2. Select **Row Height**.
3. In the text field, enter the number of pixels??. In-Sight resizes the rows.

► **To Set the Column Width**

1. Select a column.
2. Click and release O, opening the Edit menu.
3. Select **Col Width**.
4. In the text field, enter the number of pixels??. In-Sight resizes the columns.

COGNEX In·Sight

◄ ▲ ► Worksheet: Customizing

You can enable or disable worksheet elements such as the typically to implement an operator interface.

► **To Toggle Worksheet Elements**

1. Click and release the O button, opening the Edit menu.
2. Select **Worksheet**, opening the **Operator View** dialog:

Sets position of
displayed WS pane

Toggles Formula Bar

Toggles column header

Toggles row header

Toggles WS grid

Restricts scrollable
area
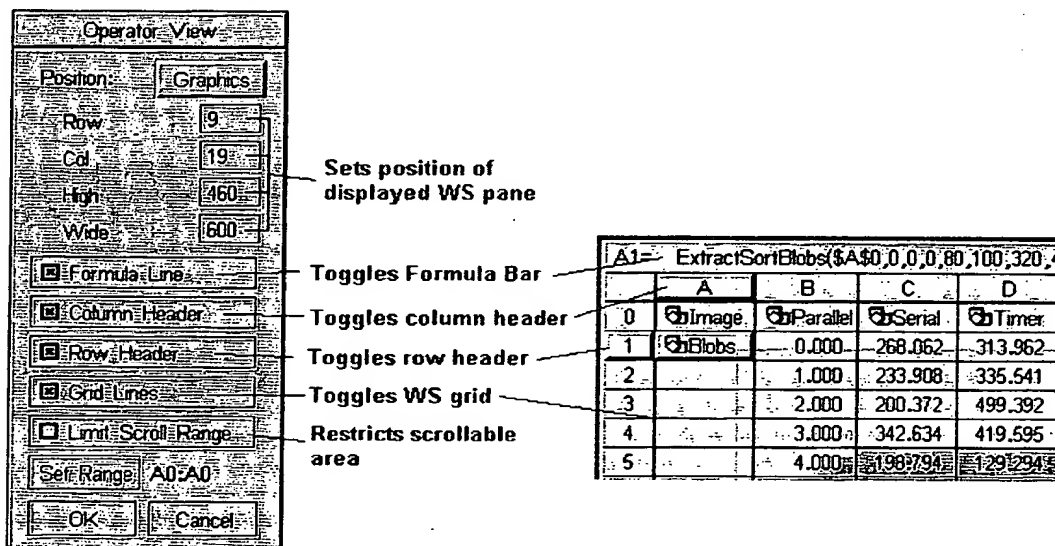
3. Enable or disable the desired attribute.

COGNEX In·Sight

◄ ▲ ► Formulas: Building Formulas

The In-Sight's worksheet interface lets you set up machine vision applications. This section discusses the following topics:

- Using the In-Sight Worksheet
- Entering a Formula
- Placing a Value on the Worksheet
- Creating an Absolute Reference
- Creating a Relative Reference
- Creating a Mixed Reference
- Using Interactive Reference Mode
- EXPRESSION -- discusses this menu item
- Opening the Property Sheet from a Formula
- Formula Syntax
- Math Functions and Operators
- Dealing with Errors.

A *cell* is a location in the In-Sight worksheet identified by its row and column address. Each cell can hold a value, a string, an expression, or a structure. A value is a literal quantity, such as 1.00. A string is one or more alphanumeric characters surrounded by quotes, such as "string". An expression is a formula that evaluates to a value or string; the worksheet shows the result. A structure encapsulates complex data, such as an image. When you create a structure, it often emits some of its data to the following cells through "getter" functions; alternatively, you can reference the structure from a getter.

COGNEX In-Sight

◄ ▲ ► Formulas: Placing in the Worksheet

The worksheet, where the real work gets done, is a simple spreadsheet. Like standard spreadsheets, the In-Sight worksheet is a table of cells with numbered columns (1 through 99) and lettered rows (A through Z). Each cell can have an associated formula (an expression), and the floating-point result of the formula is the cell's value. The worksheet displays the active cell's coordinate (here, A1) in the upper-left corner, its formula (6-5) in the formula line, and its value (1.000) in the cell. This layout will be natural to anyone familiar with standard spreadsheets.

Image structure, stores acquired image

Current cell

Worksheet columns

Formula bar
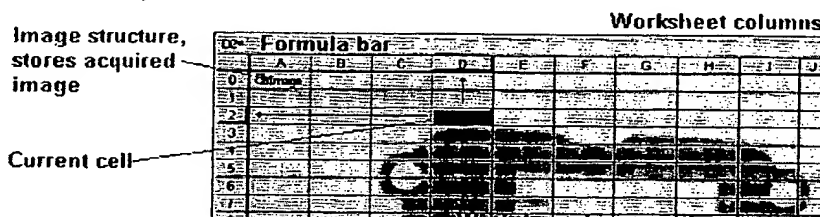
You navigate from cell to cell with the cursor. At each new cell, In-Sight updates the active cell and the formula line to reflect the current cell.

COGNEX In-Sight

◄ ▲ ► Formulas: Entering

All of the vision processing software is implemented as functions used inside formulas within a worksheet. Constructing formulas therefore is a key aspect using the In-Sight SW. To simplify the task of creating and editing formulas, In-Sight offers dialog box called *Formula Builder*, shown below.
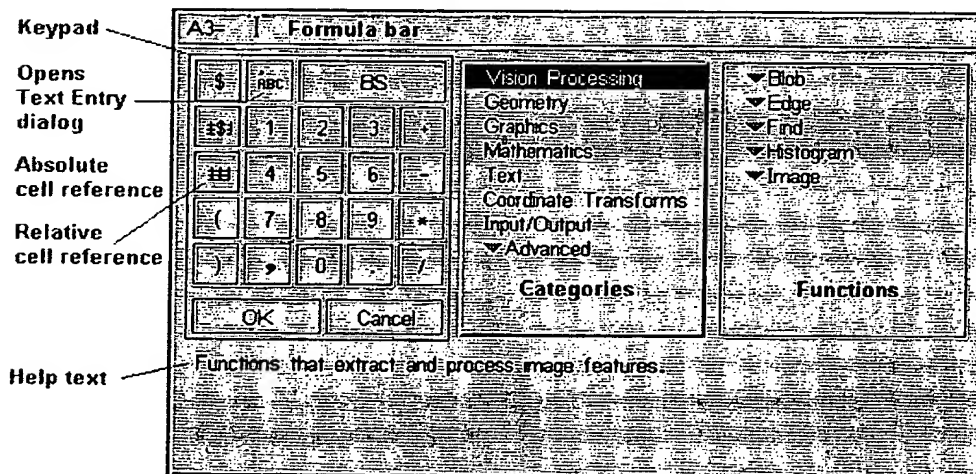
To open the Formula Builder, select a cell and click the ✕ button. The resulting dialog has three panes. The left pane offers a keypad, arithmetical operators, navigational controls, and other elements. The center and right panes offer hierarchical access to vision processing functions and to more general mathematical and spreadsheet functions.

You manipulate the cursor to move around inside the Formula Builder to select constants, operators, and functions. In-Sight indicates selectable items in blue, unselectable items in black, and the current selection in

red. Clicking ✕ inserts the current selection in the formula. For example, to insert the value "6" in cell A1's formula, select the cell A1, click ✕ to open the Formula Builder, use the cursor to navigate to the digit 6 in the keypad, and then click ✕. As shown below, Formula Builder then inserts the value "6" in the formula.

Constructing a more sophisticated expression is simply a matter of repeating this basic sequence until all of the terms are present.

COGNEX In-Sight

◄ ▲ ► Formulas: Placing a Value on the Worksheet

The simplest formula is a static value, which remains constant when the worksheet updates.

▶ **Placing a Value on the Worksheet**

There is more than one way to do this.

1. With the Control Pad, highlight the cell that will contain the value.
2. Click and hold the ✕ button, opening the Enter menu.
3. Select **Value**. In-Sight hides the Enter menu and shows the worksheet.
4. To select the digit to edit, click Left or Right.
5. To increment the value, click Up. To decrement it, click Down.
6. When you reach the desired value, click ✕ to accept it.

COGNEX In-Sight

◄ ▲ ► Formulas: Create an Absolute Reference

An *absolute reference* is a cell reference that that remains unchanged when copied to a new cell. For example, if cell **D1** contains a formula with an absolute reference to **A1**, and you copy D1 to E1, then the new formula in **E1** contains the same absolute reference to A1. With In-Sight, you can create an absolute reference from the Formula Builder, the Enter menu, or by hand.

▶ **To Create an Absolute Reference from the Formula Builder**

1. With the Control Pad, highlight the cell that will contain the reference.
2. Click and release the ✕ button, opening the Formula Builder dialog.
3. Click [▦]. In-Sight hides the Formula Builder and shows the worksheet.
4. Highlight the referenced cell, and then click ✕ to accept the reference. In-Sight returns to the Formula Builder.
5. When done with the formula, click **OK**.

▶ **To Create an Absolute Reference from the Enter Menu**

1. With the Control Pad, highlight the cell that will contain the reference.
2. Click and hold the ✕ button, opening the Enter menu.
3. Select **Absolute**. In-Sight hides the Enter menu and shows the worksheet.
4. Highlight the referenced cell, and then click ✕ to create the reference.

▶ **To Create an Absolute Reference from an Existing Reference**

1. Using the Control Pad, highlight the cell containing an existing reference.
2. Click and hold the ✕ button, opening the Enter menu.
3. Select **Formula**. In-Sight opens the Formula Builder with the existing reference in Formula Line, which is is pre-selected and ready for editing.
4. Click Left until the insertion point precedes the cell row value.
5. Click Down to enter the keypad, highlight the $ button, and click ✕ to insert the $ symbol that indicates an absolute reference.
6. Click Up to enter Formula Line, move the insertion point in front of the column value, and similarly edit a $ symbol in front of the cell column value.
7. Click OK to accept the edited reference.

See also *Creating a Mixed Reference* and *Creating a Relative Reference*.

=**COGNEX** In-Sight

◀ ▲ ▶ Formulas: Create a Relative Reference

A *relative reference* is a cell reference that changes when copied to a new location, preserving the same offset relationship. For example, if cell D1 contains a formula with a relative reference to C1 ("one above"), and you move D1 to F1, then the new formula in F1 has a relative reference to E1 ("one above").

▶ **To Create a Relative Reference from the Formula Builder**

1. With the Control Pad, highlight the cell that will contain the reference.
2. Click and release the ✕ button, opening the Formula Builder dialog.
3. Click [▦]. In-Sight hides the Formula Builder and shows the worksheet.
4. Highlight the referenced cell, and then click ✕ to accept the reference. In-Sight returns to the Formula

Builder.

5. When done with the formula, click **OK**.

▶ **To Create a Relative Reference from the Enter Menu**

1. With the Control Pad, highlight the cell that will contain the reference.
2. Click and hold the ✕ button, opening the Enter menu.
3. Select **Relative**. In-Sight hides the Enter menu and shows the worksheet.
4. Highlight the referenced cell, and then click ✕ to create the reference.

▶ **To Create a Relative Reference from an Existing Reference**

1. Using the Control Pad, highlight the cell containing an existing reference.
2. Click and hold the ✕ button, opening the Enter menu.
3. Select **Formula**. In-Sight opens the Formula Builder with the existing reference in Formula Line, which is is pre-selected and ready for editing.
4. Click Left, placing insertion point in between the $ symbol and the row or column value.
5. Click Down to enter the keypad, highlight the **BS** button, and click ✕ to remove the $ symbol.
6. If the existing reference contains a second $ symbol, click Up to enter the Formula Line, and similarly remove the second $ symbol.
7. Click **OK** to accept the edited reference.

See also *Creating an Absolute Reference* and *Creating a Mixed Reference*.

COGNEX In·Sight

◀ ▲ ▶ Formulas: Create a Mixed Reference

A *mixed reference* is a cell reference where one cell coordinate is absolute and the other is relative. When you copy a cell containing a mixed reference to a new cell, the absolute part remains unchanged but the relative part changes to preserve the same offset relationship.

▶ **To Create a Mixed Reference**

The only way to create a mixed reference is to edit a preexisting reference.

1. Using the Control Pad, highlight the cell containing an existing reference.
2. Click and hold the ✕ button, opening the Enter menu.
3. Select **Formula**. In-Sight opens the Formula Builder with the existing reference in Formula Line, which is is pre-selected and ready for editing.
4. Click Left to placing insertion point in front of the cell row or column value.
5. Click Down to enter the keypad, then edit the row and column coordinates as follows:
    o To change a relative coordinate into an absolute coordinate, highlight the $ button, and click ✕ to insert the $ symbol that indicates an absolute reference.
    o To change an absolute coordinate into a relative coordinate, highlight the **BS** button, and click

          X to remove the $ symbol.

6.   Click **OK** to accept the edited reference.

See also *Creating an Absolute Reference* and *Creating a Relative Reference*.

COGNEX In-Sight

◄ ▲ ► Formulas: Set Values Interactively

??This should discuss the interactive methods for setting cell references. When you enter this mode, transparency is OFF, and the background color is a bit darker.

Many functions have parameters that define the size, location, or angle of a geometric shape. For example, LineToCircle()
measures the distance between a line and a circle. The line is defined by four values (for both end points, a row

Often, the most convenient way to set these values is by manipulating an image of the shape with the Control P

► **Using Interactive Reference Mode**

1. Open a property sheet, and then select a heading for a parameter that can be set graphically.
2. Click and hold the ✕ button, opening the Enter menu.
3. Select ???. In-Sight hides the worksheet and displays a cursor appropriate for the parameters being set. For example, to set a circle, it displays a circular cursor.
4. With the Control Pad cursor, manipulate the graphical cursor until it reflects your needs.
5. When satisfied, click ✕, returning to the property sheet.

COGNEX In-Sight

◄ ▲ ► Formulas: Open a Property Sheet

??This topic discusses the VALUE item on the Enter menu

► **Placing a Value on the Worksheet**

There is more than one way to do this.

- xxx
- xxx
- xxx.

COGNEX In Sight

◄ ▲ ► Formulas: Syntax

??note that a vision tool can reside only at the top level. It can put its results into a cell, but not into a formula.

When you're editing a formula, you can generally ignore syntax, because Formula Builder automatically disables invalid characters, preventing you from entering them. The following figure shows the formula syntax in graphical form.
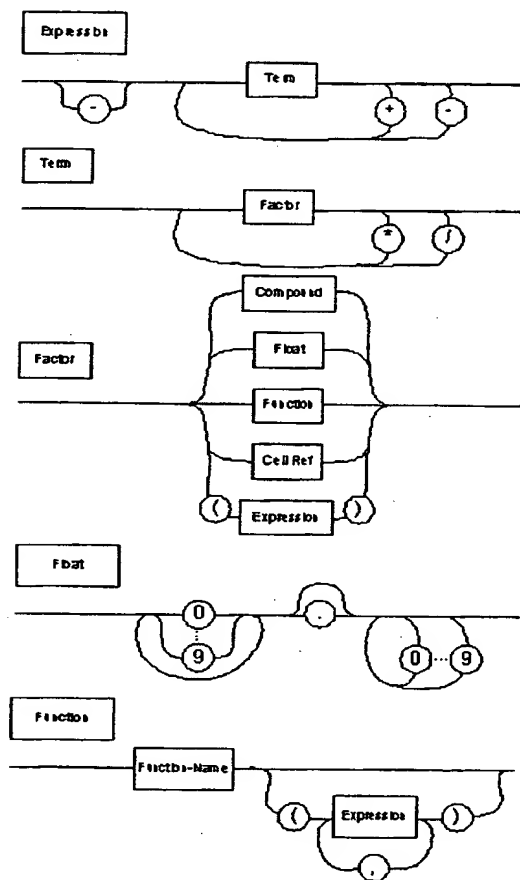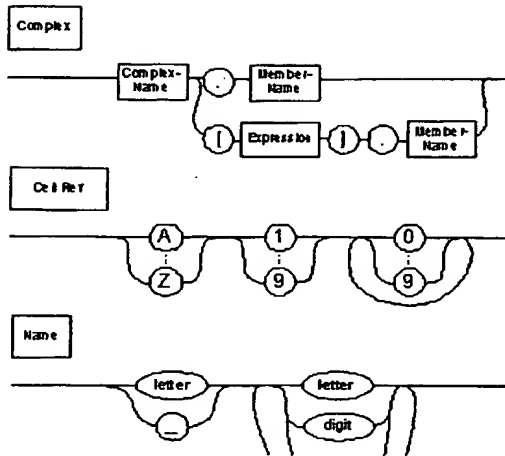
COGNEX In-Sight

◄ ▲ ► Formulas: Math Functions and Operators

In-Sight offers a full compliment of conditional, logical, arithmetical, statistical, and trigonometric functions plus standard relational operators. Rules affecting the mathematical operations include the following:

- To assure compatibility of all numeric values in the worksheet, In-Sight represents all numbers as floating-point values. For example, it represents the integer 3 as 3.00. Similarly, for logical functions, it represents TRUE as non-0.00 and FALSE as 0.00. Eliminating the need for explicit numeric data types simplifies the task of developing spreadsheets for vision applications.
- Some values are intrinsically integer. Supplying a fractional value for an integer parameter results in rounding to the nearest whole integer. For example, 3.1 rounds to 3.000, but 3.7 rounds to 4.000.
- The underlying precision (32-bit floating-point) exceeds its displayed precision (three decimal places). Computations on floating-point values reflect the internal precision.
- All indexes are zero-based.
- A reference to an empty cell observes the following rules:
  - When referenced by functions that take cell-range arguments, the empty cell is ignored. For example, if A3:A5 are all empty, then Mean(1, A3:A5, 2, 3) returns 2, the average of the valid values 1, 2, and 3. The only exception is Sum(), which treats an empty cell as 0.0.
  - When referenced as a numeric or Boolean value, the empty cell is evaluated as 0.0. For example, if A1 is empty, then the formula 3.00 + A1 evaluates to 3.00. Similarly, Abs(A1) returns 0.0.
  - When referenced as a string, the empty cell is evaluated as a null-string. For example, if A1 is empty, then Concatenate(A1) returns a null (*empty*) string.
- Error propagation:

- And(), Or(), Sum(), Max(), Min(), Mean(), and SDev() ignore individual parameters that refer to error cells. If all input parameters are #ERR, then the result is #ERR. For example, Sum(2, #ERR, 2) yields 4, but Sum(#ERR, #ERR) yields #ERR.
- MaxI() and MinI() assign indexes to all parameters that refer to error cells, but do not consider them when evaluating a result. For example, MaxI(0, 1, 2, #ERR, 15) returns 3.00, the zero based index of 15.

For more information, see *Mathematics Functions*.

COGNEX In-Sight

◀ ▲ ▶ Formulas: Dealing with Errors

On success, In-Sight functions return a value or a string. On failure, they return one of the values in the following table.

Error	Description
#PARM	Indicates an invalid parameter value. Generally occurs only when passing a value by cell reference, since the Formula Builder performs range checking.
#ERR	Indicates a general error.
#TIMEOUT	Indicates that a function timed out during execution. ??Can occur only if the timeout feature is enabled by XXXX.

COGNEX In-Sight

◄ ▲ ► Vision and Worksheet

This section describes how to use the Formula Builder for vision processing:

- Default Acquisition References
- Defining Regions of Interest
    - o Defining Rectangular ROIs
    - o Defining Curved, Circular, or Toroidal ROIs
- Working with Fixtures
    - o Image, Fixture, and Region Coordinate Systems
    - o Fixture Example
- Image Processing
- Profiling Execution.

COGNEX In-Sight

◄ ▲ ► Vision: Default Acquisition References

When the In-Sight software starts up, the worksheet contains some default references, as shown below:

A0=	AcquireImage(0,0,0,167,0,0)			
	A	B	C	D
0	Image	Parallel	Serial	Timer
1				

| Stores acquired image | Stores parallel port settings | Stores serial port settings | Stores a timer |

COGNEX In-Sight

◄ ▲ ► Vision: Region of Interest

A region of interest (ROI) is an image area within which a function performs an image processing, feature extraction, or other operation. An ROI is commonly rectangular but sometimes angled, curved, circular, or toroidal. You define a rectangular ROI by its top-left point, its height and width in pixels, and its angle in counterclockwise degrees:
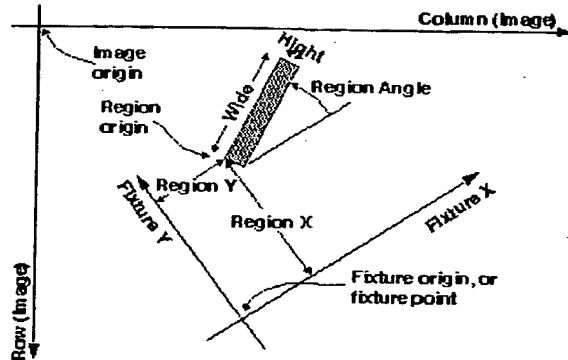
The ROI center is usually the expected feature center. The ROI size depends on positional and angular uncertainty. Keeping the ROI as small as practical reduces the number of pixels to process, reducing processing time. In-Sight supports several strategies for defining ROIs during application development and deployment. In general, an ROI can be:

- **Fixed.** The ROI parameters are unchanging values, defined either by editing them directly in a property sheet or by referring to cells containing static values.
- **Interactive.** You define the ROI with a graphic cursor drawn on the image and manipulated with the Control Pad.
- **Calculated.** The ROI values derive from worksheet formulas that adjust them at run time.
- **Fixtured.** The values are defined by row, column, and angle offsets from a fixture coordinate system identified in each image. A special case of "calculated."

For procedures used to define an ROI, see Defining a Region of Interest.

_____ **COGNEX** In Sight

◀ ▲ ▶ Vision: Rectangular Regions

You can define a region of interest (ROI) interactively, manually, or by worksheet references. You typically use the interactive method during development because it gives you immediate feedback about its size, location, angle, and curvature. You define the ROI manually when you know its exact parameters and prefer to edit them as fixed numeric values. The worksheet method is common in applications that dynamically define the ROI based on computed rather than static values.

▶ **To Interactively Define a Rectangular ROI**

Dragging a interactive cursor over the image is a convenient way to define an ROI. When using the interactive Region cursor, clicking the Control Pad ○ button cycles the cursor to set location, size, angle, and (in some contexts) curvature:
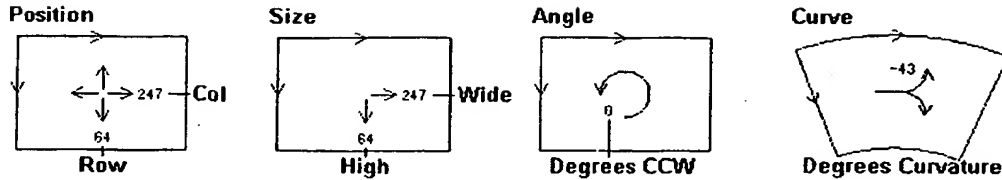
Position	Size	Angle	Curve
Col / Row	Wide / High	Degrees CCW	Degrees Curvature

The arrows on the "top" and "left" sides point away from the origin, for operations affected by the region's orientation.

1. In a property sheet, click and release the **Region** header. In-Sight hides the property sheet draws the interactive region cursor on the image:
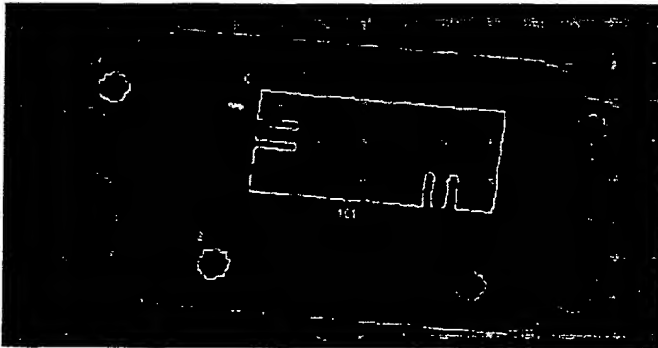
2. To adjust the position, use the Control Pad to move the four-arrowed Region cursor until its center is at the intended center. As the cursor moves, it indicates the its current row and column location.
3. To adjust the size, click O change the cursor to its two-arrowed form, then resize the ROI horizontally or vertically. Up and Down set the height, and Left and Right set the width. The cursor indicates the current height and width values.
4. To adjust the angle, change the cursor to its circular form, then adjust the angle, indicated in counterclockwise degrees.
5. To adjust the curvature (when available), set the cursor to its curved form, then adjust the curvature.
6. When satisfied with the ROI, click the △ button to accept it. In-Sight returns to the property sheet.

A region cannot extend outside the image. If the cursor hits an edge, it stops.

▶ **To Manually Define an ROI**

1. Open a property sheet and select its **Region** heading, if present.
2. Click and hold the ✕ button, yielding the **Enter** menu.
3. Click **Cursor-Down** to select **Expand** from the Enter menu. In-Sight opens the Region heading, showing the Row, Column, High, Wide, Angle, and (if available) Curve values.
4. Select a Region parameter and edit its values.
5. When done, click OK or Run to accept the values.

▶ **To Define an ROI by Worksheet Reference**

1. Create cells to serve as source of the region parameters—Row, Column, High, Wide, Angle, and (if applicable) Curve.
2. Open a property sheet and select its **Region** heading, if present.
3. Click and hold ✕, yielding the **Enter** menu.
4. Click **Cursor-Down** to select **Expand** from the Enter menu. In-Sight opens the Region heading, showing its values.
5. Select one of the Region parameters. For example, select **Row**.
6. Again click and hold ✕ to open the **Enter** menu.
7. Select Relative or Absolute, depending on the kind of reference desired. In-Sight displays the worksheet, where you select a cell as a reference source.
8. Click ✕ to accept the reference.
9. Repeat for the other Region parameters.

If you specify one of the ROI parameters by worksheet reference, then the worksheet value overrides the value set manually or interactively.

COGNEX In-Sight

◀ ▲ ▶ Vision: Curve, Circle, Torus

In some contexts, a region can be curved, circular, or toroidal (doughnut-shaped) instead of rectangular. This topic describes the interactive methods for defining non-rectangular regions.

▶ **To Define an Curved ROI**

Defining a curved ROI is similar to defining a rectangular ROI, except for the additional curvature value:
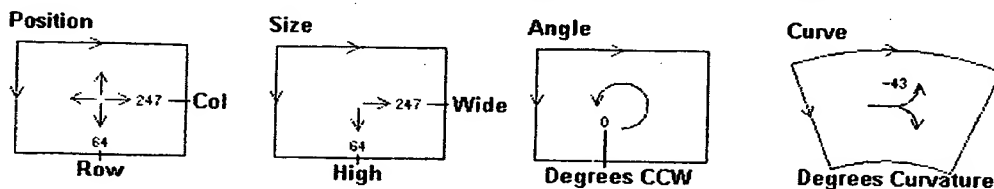
1. In a property sheet, click and release the **Region** header. In-Sight hides the property sheet draws the interactive region cursor on the image:

2. Adjust the position, size, and angle.
3. Adjust the curvature. ??Show this??

▶ **To Define a Circular ROI ROI**

You define a circular or toroidal ROI through a center point, an inner circle, and an outer circle.

1. In a property sheet, click and release the **Torus** header and click ✕ In-Sight hides the property sheet and displays an interactive cursor for defining circular or toroidal ROIs:

    This cursor has two forms, and the ◯ button toggles between them.
2. Adjust the location of the of the center point.
3. Move the center of the cursor to the center of the desired region.
4. Adjust the inner circle with the Up and Down buttons.
5. Adjust the outer circle with the Left and Right buttons
6. Click ✕ to return to the property sheet.

A region cannot extend outside the image. If the cursor hits an edge, it stops.

COGNEX In-Sight

◀ ▲ ▶ Vision: Working with Fixtures

A Fixture/ is a reference coordinate system (row, column, and angle) identified in an image and subsequently used to define other points or regions through offsets. A typical application first identifies the fixture origin and angle through pattern matching, blob analysis, or some other technique. It then stores the fixture row, column, and angle values and uses it as a benchmark within the image. Other In-Sight functions subsequently refer to the fixture point through its Fixture row, column, and theta values.

This section introduces fixtures, discussing the following topics:

- Image, Fixture, and Region Coordinate Systems
- Fixture Example

Defining locations in fixtured coordinates instead of image coordinates can improve robustness by compensating for positional and angular uncertainty. It can also improve speed by allowing many subsequent operations to use smaller ROIs, angular tolerances, and so on. Another advantage is that distances measured in the fixtured coordinate system often apply more directly to the real-world item of interest.

1.The name "Fixture" refers to mechanical forms or devices that hold parts in place for some machining or

manufacturing operations.

===== **COGNEX** In-Sight

◄ ▲ ► Vision: Coordinate Systems

In-Sight employs a right-hand coordinate system, with the origin at the top-left of the image. The reference point—located in the image by feature extraction or another method—is identified by its row and column locations, in image coordinates, and by a counterclockwise angle. The following figure shows the Image and Fixture coordinate systems:
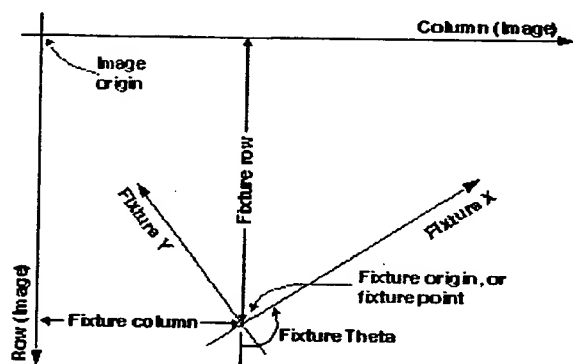
You define an ROI through its row, column, and angle offsets relative to the Fixture coordinate system, as shown below:
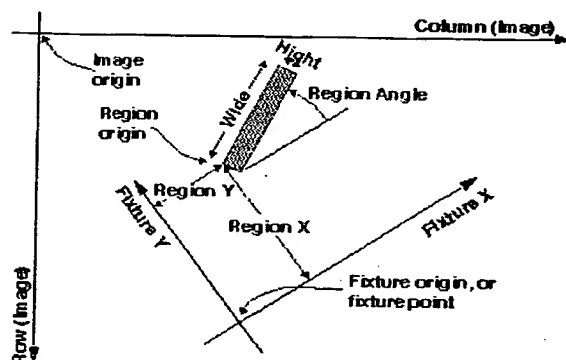
For a rotated region, width is the arc distance along the center line. Rectangular regions can be viewed as having an infinite angle:
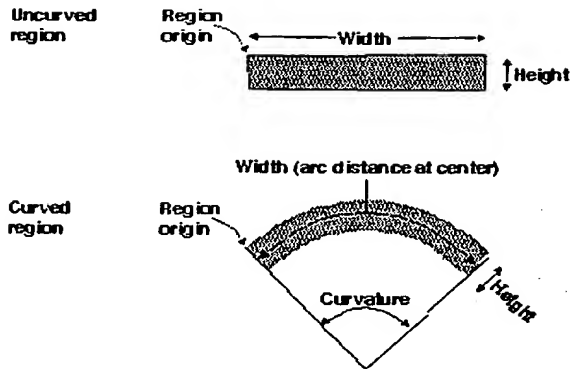
COGNEX In-Sight

◄ ▲ ► Vision: Fixture Example

?Work in an example that looks like this: You can view a search (for example) as a two-step process. Finding the fixture origin and angle amounts to preliminary search for a major landmark in the image. Based on the preliminary Finding a feature of interest amounts to a second search that depends on the preliminary values. This hierarchical strategy often greatly reduces the risk of confusion or error because the "real" search can largely ignore positional and angular uncertainty, which are absorbed by the fixture point.

Consider, for instance, an application that checks the fill level in a bottling line.

1. Find the shape of the bottle though blob analysis.
2. Set the Fixture origin to the blob's centroid and the fixture theta to its major axis.
3. Define an ROI by offsets from the Fixture origin.
4. In the ROI, detect an edge parallel to the Fixture Y axis. Applying the feature extraction in a fixtured ROI instead of the general image often greatly reduces the risk of confusion with background features that happen to resemble the feature of interest.
5. Measure the X distance from the edge to the Fixture origin. Because this measurement occurs in the Fixtured coordinates, it is unaffected by angular variation in the image.
6. Compare the measured distance to some pass-fail standard.

COGNEX In-Sight

◄ ▲ ► Vision: Image Processing

??All this has changed??

Appendix I - Page **95**

Many vision applications enhance the image with filters or other image processing operations before extracting features from it. You specify a sequence of image-processing operations ??? function's Script parameter:
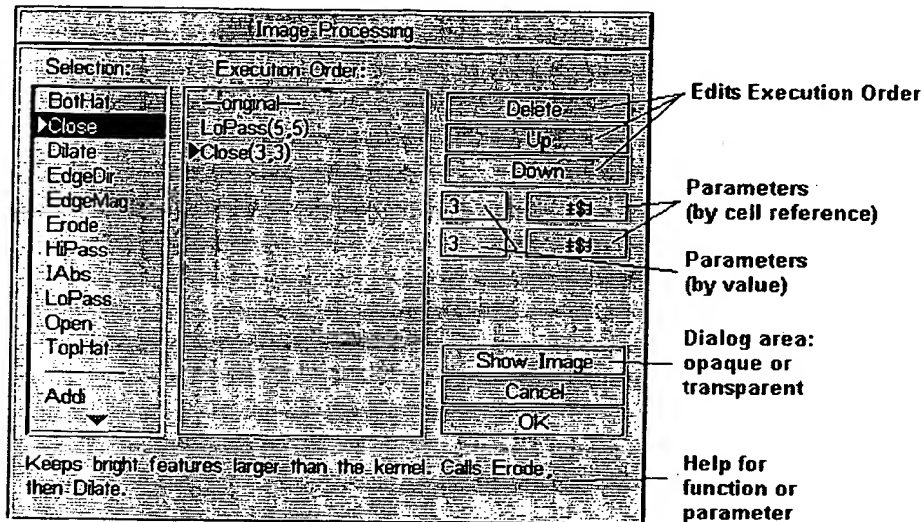
- To add an operation, highlight it in the Selection list and click X. The new item appears in the Execution Order list immediately after the item currently selected there.
- To change the execution order, select an item in the Execution Order list and click **Up, Down,** or **Delete.**
- To view the results of the image processing operations, scroll in the Execution Order pane. At each step, TBD updates the ROI to show the cumulative affect of the script up to that point.
- If the ROI overlaps with the dialog, it is sometimes hard to read the dialog when displayed at the same time as the image. To suppress transparency, click **Opaque;** to return to simultaneous display, click **Show Image.**

COGNEX In-Sight

◄ ▲ ► Vision: Profiling Execution

The Profiler gives you information about execution times within the worksheet, to help you optimize it.

► **To Profile Execution**

1. Click and release the △ button, opening the System menu.
2. Select **Profiler** and then click X. In-Sight displays the Profiler dialog:

Cell	Result	Time	State	Expression
A0	Pass	0.00ms	☒	AcquireImage(0,0,0,167,0,0
A1	Pass	36.90ms	☒	ExtractSortBlobs(A0,0,0
B5	4.00	0.01ms	☒	4
B4	3.00	0.01ms	☒	3
B3	2.00	0.01ms	☒	2
B2	1.00	0.01ms	☒	1
B1	0.00	0.01ms	☒	0
D0	Pass	0.02ms	☒	Timer(250,0)
C0	Pass	0.02ms	☒	AcquireSerial(0,4,0,0,1,0,1
B0	Pass	0.02ms	☒	AcquireParallel(1,1,1,1,1,1
C1	267.97	0.02ms	☒	GetRow($A$1,B1)
D1	313.91	0.01ms	☒	GetCol($A$1,B1)
E1	86.41	0.01ms	☒	GetAngle($A$1,B1)
F1	0.00	0.01ms	☒	GetColor($A$1,B1)
G1	0.00	0.01ms	☒	GetScore($A$1,B1)
H1	67952.00	0.01ms	☒	GetArea($A$1,B1)
I1	0.02	0.01ms	☒	GetElongation($A$1,B1)

Execute All
Execute From   Start at selected
Exit

☐ Objects Only   Limits list to visic
☐ Sort by Time   Execution time, I:
☐ Dependencies   Lists current cell

Total   37.7   ms

3.   ??What next??

COGNEX In Sight

◄ ▲ ► Options: Introduction

??You set up various system options thorough dialog boxes opened from the System menu. This section details the options, discussing the following topics:

- Setting up a Serial Port
- Setting up Discrete In
- Setting up Discrete Out
- Setting the Password
- Setting Startup Options
- Customizing the Interface
- Setting Advanced Options
- Saving the Settings
- ??Error Handling
- Getting the In-Sight Version Number
- Obtaining Online Help

COGNEX In-Sight

◄ ▲ ► Options: Setting Up the Serial Port

If your application uses serial communications, you must define the properties of the serial port

► **To Set Up the Serial Port**

1. Click and release the △ button, opening the System menu.
2. Select **Options**, opening the Options menu.
3. Select **Serial Port**, opening the Serial Port dialog.
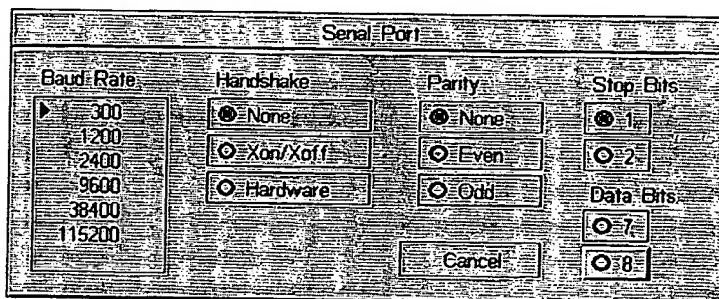
4. Set the values applicable to your serial device.

COGNEX In-Sight

◄ ▲ ► Options: Setting up Discrete Input

In-Sight offers up to 10 discrete inputs and 10 discrete outputs. In hardware, each line is a hot/ground pair of screw terminals used to connect photocells, pressure sensors, warning lights, and other I/O devices. The external inputs are numbered 0 through 7, to agree with the labels printed on the I/O Expansion Module. The built-in inputs are numbered 8 and 9.

► To Set up Discrete In

1. Click and release △, opening the System menu.
2. Select **Settings** and then click ✕, opening the Settings menu.
3. Select Discrete Input, yielding the Discrete Input dialog:

4. Click **Details**:

The parameters are:

o Name. A text label.
o Debounce time. Allows In-Sight to ignore changes in the signal for a specified number of milliseconds after a transition, preventing it from misinterpreting noise as a new transition. Mechanical relays, for example, tend to produce noisy signals that require debouncing.
o Invert polarity. When OFF (the default), polarity is active-high, so the transition from 0v to 12v DC indicates ON. When ON, polarity is active-low, so the transition to 0v DC indicates ON.

COGNEX In-Sight

◄ ▲ ► Options: Setting up Discrete Output

??This one talks about discrete out?? The external outputs are numbered 0 through 7, to agree with the labels printed on the I/O Expansion Module. The built-in outputs are numbered 8 and 9.

► **To Set up Discrete Out**

1. Click and release △, opening the System menu.
2. Select **Settings** and then click ✕, opening the Settings menu.
3. Select **Discrete Output**, yielding the Discrete Output dialog:

4. Click **Details**. You get:

The parameters are:

o Name. A text label.
o Forcing.
o Invert polarity. When OFF (the default), polarity is active-high, so In-Sight switches from 0v to 12v DC to indicate ON. When ON, polarity is active-low, so the transition to 0v DC indicates ON.
o Output scheduling
o Delayed
o Pulsed output

COGNEX In-Sight

◄ ▲ ► Options: Setting the Password

??all of this has probably changed?? The In-Sight software has one level of password protection, "unlocked" versus "locked":

- When locked, the operator can view the image but cannot view or change the job settings. When locked, pressing any button yields the Unlock dialog. The only way "in" is to enter the correct password, a user-assigned string.
- When unlocked, the operator can view or change any parameter. When unlocked, the user can lock the software by selecting Lock, yielding the Lock dialog, shown below. After supplying and confirming a password, In-Sight toggles to "locked."

There is no password dialog as such. Locking the system always requires entering a password twice. The new password might or might not match the previous password. The password could potentially change every time the software is locked; unlocking always requires the previous password.

The menu label changes according to its state. ?The software can be locked or unlocked at startup, depending on an option in the Options dialog.?

▶ **To Lock the System**

1. Click and release the △ button, opening the System menu.
2. From the System menu, select **Lock** and then click ✕. In-Sight displays the Lock dialog:
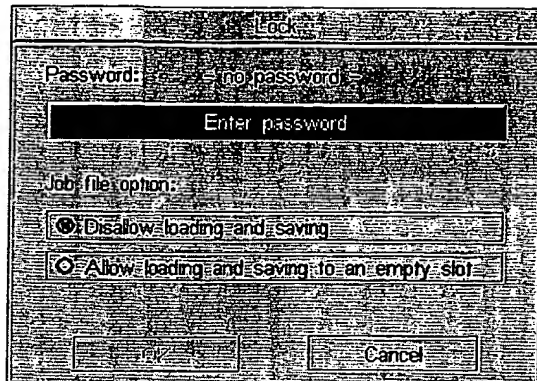
3. ??What else??

<hr>

COGNEX In Sight

◄ ▲ ► Options: Setting Startup Options

If your application uses serial communications, you must define the properties of the serial port

▶ **To Set Up Startup Options**

1. Click and release the △ button, opening the <u>System menu</u>.
2. Select **Settings**, opening the Settings menu.
3. Select **Startup**, opening the Startup dialog.
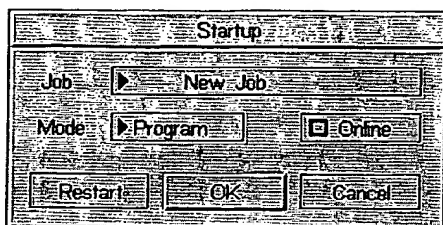
4. Set the applicable values.

<hr>

COGNEX In Sight

◄ ▲ ► Options: Language and Color Scheme

In-Sight collects general system options in the Customize Dialog. This state is written to flash memory on an ongoing basis.

► **To Set an Option**

1. Click and release the Δ button, opening the System menu.
2. Select **Options**, opening the Options menu.
3. Select **Customize**, opening the Customize dialog.
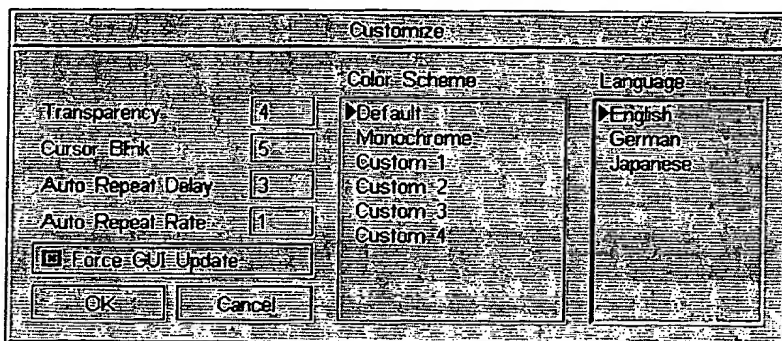
4. Select the desired options, described below:

Option	Description
Language	The language used for the built-in help text. If you change the language and can't get back: (1) Click □ to open the System menu; (2) Select the last item in the System menu and click ✗, opening the Options menu; (3) Select the first item and click ✗, opening the Customize dialog (4) Click **Cursor Left** to activate the Left pane; and (5) Select a language you can read.
Transparency	Controls the blend between the image and the worksheet for simultaneous display. The valid range is 0 to 100, where small values lighten the image and large values darken it.
Cursor Blink	Sets the rate of the blinking highlight that indicates the current selection, in milliseconds. The valid range is 0 (off) to 9999 (almost ten seconds).
Auto Repeat Delay	Defines the duration of a single click for the Control Pad, in milliseconds. Holding down a button beyond this value cause it to automatically repeat the keystroke. The valid range is 0 (off) to 9999 (almost ten seconds).
Auto Repeat Rate	Sets the duration of each keystroke automatically generated after holding down a button beyond the Auto Repeat Delay. The valid range is 0 (off) to 9999 (almost ten seconds).
Color Scheme	Controls the overall appearance of the graphical interface. In-Sight draws the worksheet and other GUI elements on top of the acquired grayscale image. It depends on color to separate the GUI from the image.

The following table shows the default color conventions.

Color	Meaning
Light yellow	The background color for menus, dialog boxes, and so forth.
Blue	"Hot" text, potentially selectable. The primary color contrast is blue-on-yellow.
Red text, flashing blue	The current position in a menu or dialog. A second cue for "current position" is the 3D highlighting applied to the current item.
Black	Unselectable text. Equivalent to "grayed out" in many GUIs.

COGNEX InSight

◄ ▲ ► Options: Setting Advanced Options

??Tells about the Advanced Options

► **To Set Advanced Options**

1. Click and release the △ button, opening the System menu.
2. Select **Settings**, opening the Settings menu.
3. Select **Advanced**, opening the Advanced dialog:
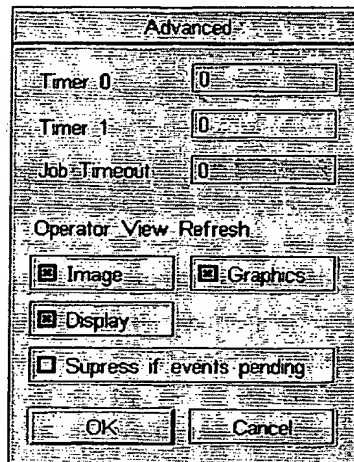
4. Set applicable values.

COGNEX InSight

◄ ▲ ► Options: Saving Settings

Settings are ??what??:

▶ **To Save the Settings**

1. Click and release the △ button, opening the <u>System menu</u>.
2. Select **???** and then click ✕. In-Sight ???.
3. Click △ to close the System menu.

COGNEX In-Sight

◀ ▲ ▶ Options: Setting up Error Handling

??This stuff moved to discrete I/OYou use the Error Handling dialog to control how In-Sight responds to errors in a production environment.

▶ **To Set Up Error Handling**

1. Click and release △, opening the <u>System menu</u>.
2. Select **Error Handling** and then click ✕, opening the Error Handling dialog:
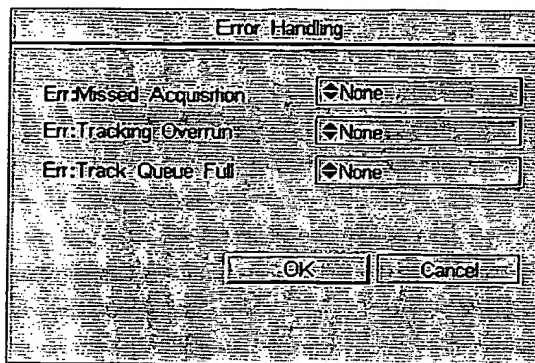
The parameters are:

- o Missed acquisition. None, Go Offline, or Reboot. ?Maybe this should be reset??
- o Tracking overrun. None, Go Offline, or Reboot.
- o Track Queue full. None, Go Offline, or Reboot.

COGNEX In-Sight

)

Appendix I - Page **105**

◄ ▲ ► Options: Software Version

??Talks about the Version option on the System menu.

COGNEX In-Sight

◄ ▲ ► Options: Online Help

In-Sight offers built-in help messages displayed in the Formula Builder, the property sheets, and elsewhere in the interface. The messages aim to provide a basic description of each function or parameter in context.

??Will we offer context sensitivity to this CHM file??

COGNEX In-Sight

◄ ▲ ► Operator Interface: Building

In-Sight offers several features for rearranging and reformatting the worksheet, for instance to simplify the view presented to an operator. Resources for this step include:

- Hiding rows or columns, to simplify the worksheet.
- Customizing the worksheet, enabling or disabling its grid lines, row headers, column headers, expression line, scrollable area, and other attributes.
- Creating graphics for user input through Button, CheckBox, EditInt, or ListBox.
- Creating graphics for user output through Status, which colors a cell green, yellow, or red, according to its value.
- Creating charts for user output.
- Locking the job, to prevent unauthorized change.
- Disabling individual cells, to prevent unauthorized change.
- Changing the color scheme.

These mechanisms let give you considerable control over system as viewed by the operator or other end user.

COGNEX In-Sight

◄ ▲ ► Server: About

The In-Sight server (isserver.exe) is Microsoft Windows NT program that accompanies the In-Sight hardware and communicates with it over a serial line. Its general behavior is similar to other MS Windows programs. To use it, you select commands from menus using the mouse, through keyboard shortcuts, or by clicking icons in the toolbar:
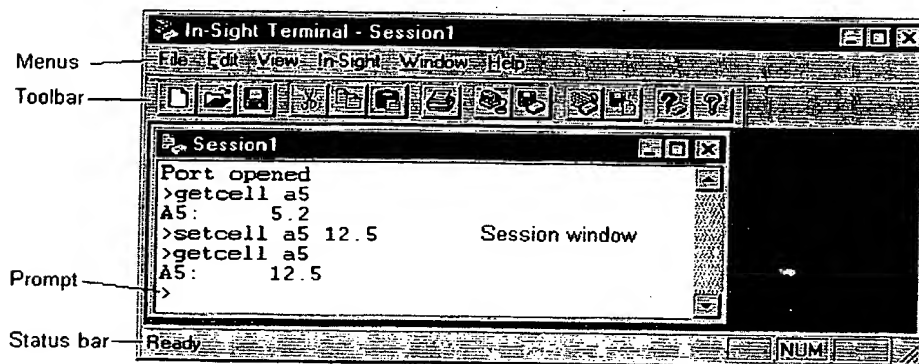
Its capabilities include:

- Issuing remote commands to In-Sight (acquire an image, read cell values, write cell values, and so on).
- Archiving jobs, images, or settings from the In-Sight hardware to a PC.
- Keyboard control, bypassing the Control Pad.
- New sessions or job files ??what is this about??
- Saving the terminal session to a file for later review.

This section describes the server's In-Sight-specific behavior, discussing the following topics:

- Configuring the Server   • Running In-Sight by Keyboard
- Issuing Server Commands   • Server Commands Reference.
- Saving Jobs, Images, and Settings

To install the server program, see *Installing the Software*.

_____ **COGNEX In-Sight**

◄ ▲ ► Server: Configuring

To run the In-Sight server, you configure the In-Sight software and the server software with compatible communications settings.

▶ **To Configure the In-Sight Server**

1. If necessary, <u>connect</u> the Vision Processor to a COMM port on the PC with a serial cable and <u>install</u> the Server program.
2. <u>Start up</u> the In-Sight hardware.
3. Configure one of In-Sight's serial ports:
   a. Click △ to open the System menu.
   b. Select **I/O Settings** and click ✕ to open the I/O Settings menu.
   c. Select either Port 0 or Port 1 and click ✕ to open the Port dialog.
   d. Set **Mode** to Server and the **Status** to Open:
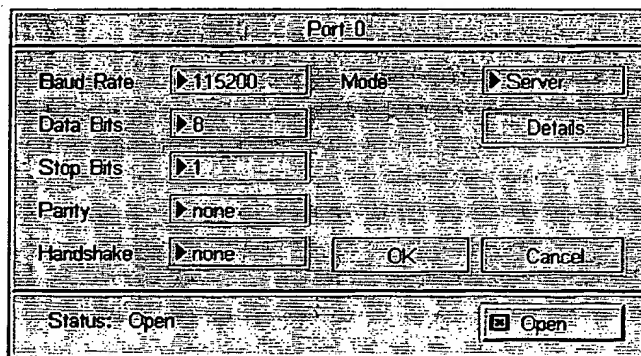
   In Server mode, inapplicable options are grayed out.

   e. Click OK to accept.
4. Start the In-Sight server program, **isserver.exe**.
5. Configure the server to agree with the COMM port in use:
   a. Click 📷 or select Port Settings from In-Sight menu. The server opens the Port Settings dialog:
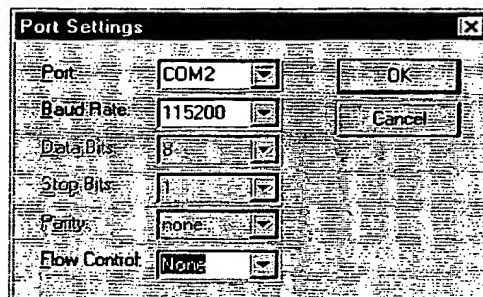
   b. Set **Port** to COMM1, COMM2, COMM3, or COMM4, depending on the port in use on the PC.
   c. Set the baud rate to ???

Appendix I - Page 109

d. Click OK to accept the changes or Cancel to close. On success, the server issues a Port Opened message.

To exit the server program, select Exit from the File menu.

COGNEX In-Sight

◄ ▲ ► Server: Issuing Server Commands

The In-Sight server can issue commands to the In-Sight hardware over a serial connection. For example, you can trigger an acquisition, read cell values, or write cell values (including valid formulas):
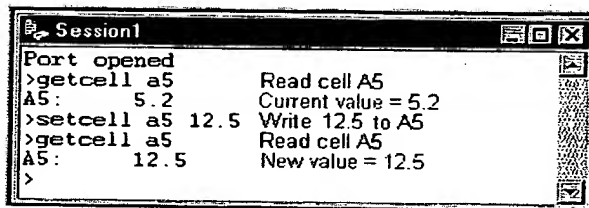
The In-Sight server generally behaves like most terminal programs: you type commands and their arguments at a prompt and view the data returned.

► **To Issue Serial Commands to In-Sight**

1. <u>Start and configure</u> the In-Sight server program and hardware.
2. At the prompt, enter a valid command:
   - Formulas passed as arguments to the **setcell** command are entered into a cell exactly as if they were entered through the Control Pad. For example, setcell A1 Abs(-25) places the expression Abs(-25) in cell A1, where it is evaluated to 25.
   - Cell references and formulas are case-insensitive.

   For command syntax, type **help** or see *Server Commands*.
3. Enter a carriage return. The server confirms that the command is valid. If valid, the server sends the command to the In-Sight hardware, which executes the command. For some commands, the hardware sends data or a message back to the server, which prints the text on the next line.

The toolbar lists session-related commands:
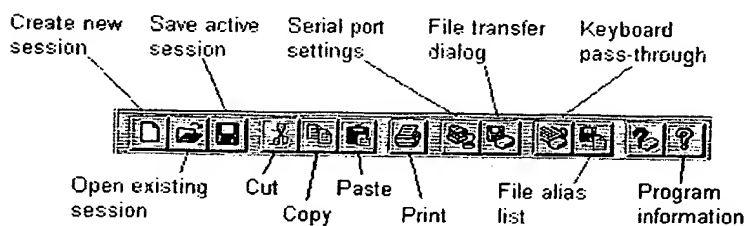
COGNEX In-Sight

◄ ▲ ► Server: Saving Jobs, Images, Settings

The In-Sight server can copy jobs, images, or settings from the hardware to the PC or from the PC to the hardware. The server therefore provides the mechanism to archive production applications and to transfer information from one In-Sight system to another.

After you have established a connection, you can initiate the transfer from either the In-Sight server or the In-Sight hardware. This topic describes how to

► **To Save Jobs, Images, or Settings**

1. Start and configure the In-Sight server program and hardware.

2. Click ▣ or select File Transfer from the In-Sight menu. The server opens the File Transfer dialog:
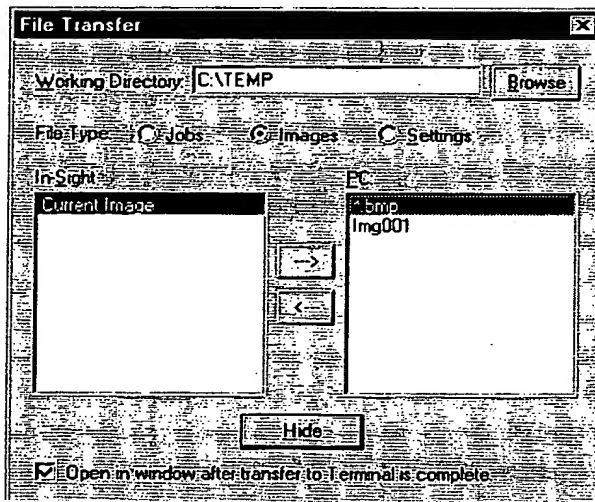
3. To select a working directory on the PC, click **Browse** and designate a directory through the resulting dialog.
4. Click Jobs, Images, or Settings to specify the type of data to transfer.
5. To transfer data from the hardware to the PC, click ▣. To transfer data from the PC to the hardware, click ▣.

NOTE
- An image file contains almost 500 KB of data. The transfer time depends on connection speed and can be lengthy for slow connections.
- Image file names are assigned sequentally starting with Img001.bmp. You cannot specify the file names.

6. When done with the File Transfer dialog, click Hide to close it.

COGNEX In-Sight

◄ ▲ ► Server: Running In-Sight by Keyboard

The In-Sight server can pass text entered on the PC keyboard to the In-Sight hardware. With this technique, you can select functions by typing their names and enter text comments from the keyboard instead of the Control Pad. In-Sight behaves as if you had connected the keyboard to it directly. You can select function by matching their names, navigate with the keyboard cursor, and so on.

► **To Run In-Sight by Keyboard**

1. Start and configure the In-Sight server program and hardware.

2. Click ▣ or select **Keyboard Pass-Through** from the In-Sight menu.

3. At the server prompt, enter a cell address followed by a carriage return. The In-Sight hardware opens the Formula Builder dialog ready to enter a string, function, or formula in the specified cell:

A: ex					Text received over serial connection	ex
$	ABC	BS				
±$	1	2	3	+	Functions starting with current text	ExtractBlobs
#	4	5	6	-		ExtractHistogram
						Exp
						Exact
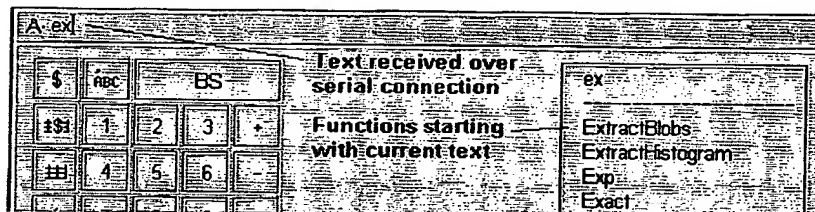
4. At the server prompt, type in an alphanumeric string. The server passes the string to the In-Sight hardware, which displays it at the top of the Formula Builder dialog. In the right-hand pane, the In-Sight lists functions that match the current string (case-insensitive). Adding letters progressively shortens the list:
   o For a unique match, In-Sight selects the matched function. Enter an open-parenthesis to delimit the parameter list, continue typing the comma-delimited parameters, and terminate the list with a close-parenthesis.
   o For no match, ??What happens??
   o For incorrect syntax, ??What happens??

5. When done with the string, enter a carriage return. In-Sight puts the finished function in the selected cell.

Keyboard usage over the the serial connection is:

In-Sight displays an image on a standard VGA monitor connected through a standard 15-pin connector.

VGA Port

VGA Display
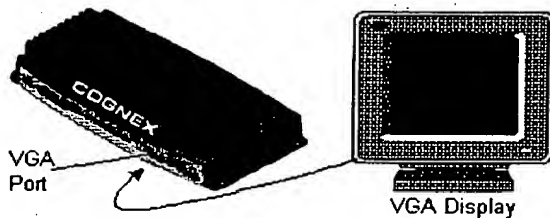
▶ **To Connect a Monitor**

1. Obtain a VGA-compatible monitor or flat-panel LCD display.
2. Remove the power from the Vision Processor.
3. Plug the monitor's video cable into the Vision Processor's VGA port.
4. Lock down the connector with its screws.

For details about the monitor signals, see *Monitor Output*.

===================================== **COGNEX In-Sight**

◄ ▲ ► Install: Serial Device

In-Sight provides two high-speed (115 kbaud) RS-232 serial ports for local communication with a PC or other devices. Serial ports 0 and 1 both use an RJ-45 connector instead of the more common DB-9 or DB-25 connector. You can use a Category 5 cable for serial communications, converting to DB-9 or DB-25 through an adapter if necessary. To accommodate DB-9 devices, In-Sight ships with an adapter (P/N XXX-YYY) that has RJ-45 on one side and DB-9 on the other.

In-Sight includes a ?10-foot? Category 5 cable for serial communications. Maximum supported cable length is ?? feet.

PC

Keyboard

COMM1, COMM2, COMM3, COMM4

Serial Cable ??with adapter??

Serial 0 or Serial 1 (RJ-45 connectors)
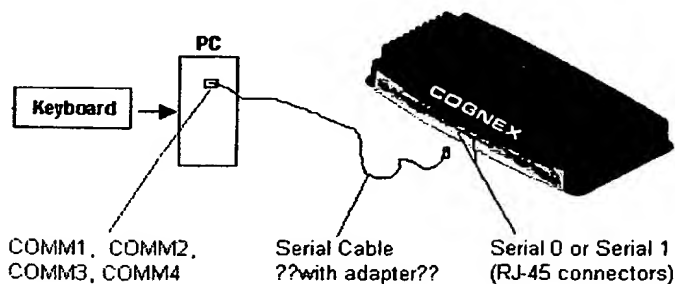
CAUTION The Vision Processor has three RJ-45 connectors, one for the camera and two for the serial ports. Make sure you plug each device into the correct connector. Plugging a serial device into the camera port can damage the serial device, the Vision Processor, or both.

▶ **To Connect a DB-9 Serial Device**

1. Remove power from the Vision Processor.
2. Plug one end of a Category 5 cable into one of the Vision Processor's serial ports, labeled SERIAL 0 and SERIAL 1.
3. Plug the other end of the cable into the RJ-45 side of the serial adapter.
4. Plug the DB-9 side of the adapter into the serial device (or, if you need more length, into a DB-9 cable).
5. To prevent strain on the connectors, fasten the serial cable to secure points near both ends, allowing some slack between the fasteners and the connectors.
6. Define the properties of the serial port though the _Serial Port dialog_.

▶ **To Connect an RJ-45 Serial Device**

1. Plug one end of a Category 5 cable into one of the Vision Processor's serial ports.
2. Plug the other end of the Category 5 cable into the serial device.
3. To prevent strain on the connectors, fasten the serial cable to secure points near both ends, allowing some slack between the fasteners and the connectors.
4. Define the properties of the serial port though the _Serial Port dialog_.

For hardware details about the serial ports, see _Serial Ports_. For details about serial communications, see _Input/Output Functions_.

COGNEX In·Sight

◀ ▲ ▶ Install: Built-In Discrete I/O

In-Sight offers a total of ten discrete inputs and ten discrete outputs for general-purpose use. Two inputs and two outputs are built in to the Vision Processor. The remaining inputs and outputs require an external _I/O Expansion Module_. Each discrete I/O signal is electrically independent from the others. All are optically isolated.

You connect each discrete I/O line to the Vision Processor through positive and negative screw terminals on an intermediate, detachable terminal block, which bundles the wires together for easy assembly and reassembly. Wiring size is 22 to 14 AWG.

Twisted pair or other wiring

PLC or other device   8-pin Terminal Block   I/O Port (8-pin Terminal)
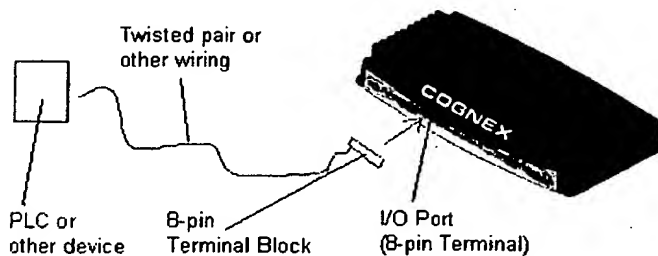
▶ **To Connect a Discrete Input or Output**

1. On the 8-position terminal block, insert the positive wire into the positive terminal and the negative wire into the negative terminal.

IN 8+   IN 9+   OUT 8+   OUT 9+
IN 8−   IN 9−   OUT 8−   OUT 9−
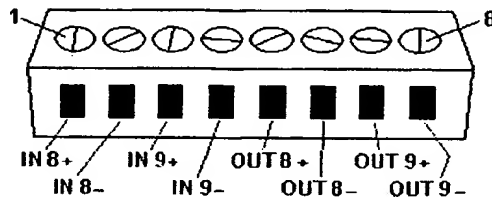
2. Tighten the set screws.
3. Similarly connect any additional discrete inputs and outputs to the terminal block.
4. Remove the power from the Vision Processor.
5. Plug the terminal block into the Vision Processor's 8-pin I/O port.
6. To prevent strain on the connectors, fasten the discrete I/O wires to a secure point near the Vision Processor, allowing some slack between the fastener and the terminal block.
7. Define the properties of each input or output as described in *Setting Up I/O Options*.

For hardware details about the I/O signals, see *Built-In Discrete I/O*.

COGNEX In-Sight

◄ ▲ ► Install: External Discrete I/O

In-Sight offers a total of ten discrete inputs and ten discrete outputs for general-purpose use. Two inputs and two outputs are built in to the Vision Processor. The remaining inputs and outputs require an optional external I/O Expansion Module (P/N XXX-YYYY). This topic describes how to connect the I/O Module to the Vision Processor and how to connect input and output lines to the I/O Module.

PLC or other device

Twisted pair or other wiring — External I/O module — External I/O cable — EXTERNAL port (26-pin connector)
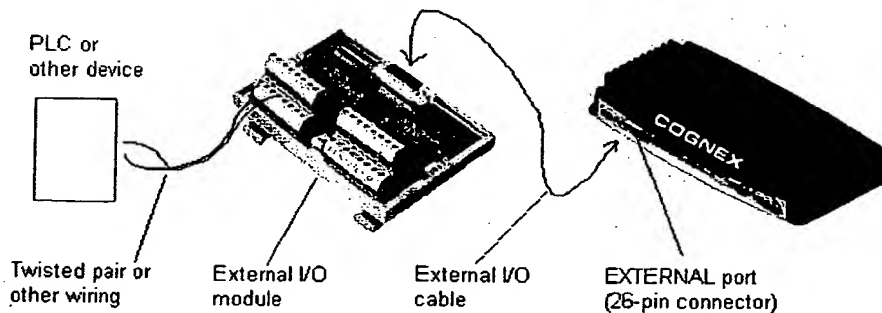
CAUTION The External I/O module must have exactly the same ground potential as the Vision Processor chassis. Any difference in potential creates a ground loop that can damage the equipment. Normally, the I/O module's housing electrically isolates it from its attachment point, allowing the Vision Processor to supply a ground through the External I/O cable. If you connect a ground wire to the I/O module's ground screw, then that ground must be electrical identical to the Vision Processor's ground.

▶ **To Connect the I/O Module**

1. Attach the External I/O Module to a convenient surface. It is configured for a standard #3 DIN rail mounting.

2. Remove the power from the Vision Processor.
3. Plug one end of the External I/O cable into the I/O Module's 26-pin connector and the other end into the Vision Processor's 26-pin EXTERNAL I/O port.
4. Connect any input lines to the input screw terminals on the I/O module and tighten the set screws.
5. Connect any output lines to the output screw terminals on the I/O module and tighten the set screws.

NOTE The Vision Processor supplies power to the I/O Module, which therefore does not require an external power supply.

COGNEX In-Sight

◄ ▲ ► Install: Power Supply

In-Sight requires a 24-volt DC power supply. The voltage tolerance is ± 5 percent, and the maximum power consumption is 1.0 Amp. You connect the power supply wires (22 to 14 AWG) to the Vision Processor through screw terminals on an intermediate, detachable terminal block, which bundles the wires together for easy assembly and reassembly.

Some In-Sight configurations include an AC adapter (P/N XXX-YYYY) that converts 100v to 240v AC (50 to 60 Hz) line current to 24v DC; other configurations assume you already have a 24v DC power source.
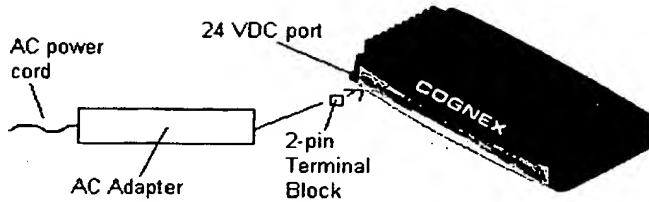
▶ **To Connect a Power Supply**

CAUTION Incorrect voltage creates a risk of fire or shock and can damage the In-Sight hardware. Never connect voltages other than 24v DC to the Vision Processor.

CAUTION Incorrect polarity can damage the In-Sight hardware. The probable result is a blown fuse, which automatically resets itself when you remove the power. The fuse is NOT user-replaceable if it does not reset itself. Always observe the polarity marked on the Vision Processor.

1. On the terminal block (Cognex P/N XXX-YYY, or Phoenix Connector #1803578), connect the positive wire to the positive screw terminal and the negative wire to the negative (ground) screw terminal:
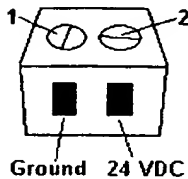
??If you are using the Cognex AC adapter, the positive wire has a white stripe and a rough texture, and the negative wire has no white stripe and a smooth texture. You can extend the three-foot DC leads as needed with suitable two-conductor wiring. If the AC plug does not match your local power service, you can remove the AC cord and replace it with a local equivalent.

2. Tighten the set screws.
3. Plug the terminal block into the Vision Processor's 24 VDC port.
4. Route the power supply wires away from sources of high-voltage power. This precaution reduces the risk of damage or malfunction due to overvoltage, noise, power surge, electrostatic discharge (ESD), and other irregularities.
5. To prevent strain on the connector, fasten the power wires to a secure point near the Vision Processor, allowing some slack between the fastener and the Vision Processor.
6. Turn the power ON. To confirm that power is present, check the LED indicator at the left side of the front panel. If the LED is dark and you believe you that power is present, one possible problem is a blown fuse. Confirm that power is present and has the correct polarity.

For power supply details, see *Power Supply Reference*.

═══════════════════════════ COGNEX In-Sight

◄ ▲ ► Install: In-Sight Software

The vision processing software is built into the In-Sight hardware and does not require installation. This topic describes how to install the server software and online documentation on a Microsoft Windows NT system.

► **To Install the Software and Documentation**

1. Before inserting the In-Sight CD-ROM in a CD-ROM drive, shut down any running applications.
2. Insert the Cognex In-Sight installation CD-ROM into your computer's CD-ROM drive.
3. Follow the setup dialogs as they appear on screen. The default installation directory for Windows NT is **C:\Program Files\In-Sight**. We recommend accepting the default.
4. When the installation program terminates, remove the In-Sight installation disk from your CD-ROM drive.
5. Test the online documentation, distributed as a compiled HtmlHelp file, by double-clicking the **C:\Program Files\In-Sight\In-Sight.chm** file. If you cannot open In-Sight.chm, then:
   a. Open Microsoft Internet Explorer and check the version number. To view the online documentation (), you must have IE 4.0 or higher. If you do not have Internet Explorer or have an older version, then update your system with a current version.
   b. If you have IE 4.0 and still cannot open In-Sight.chm, then run hhupd.exe, an update program supplied on the In-Sight CD-ROM. Early versions of IE 4.0 recognize compiled HtmlHelp (.chm) files unless they are updated. Later versions do not require updating.
6. Running the Server program requires at least one enabled serial port. If you attempt to run the server program and receive a "can't open port" error, then you must enable the port on the PC. This usually requires changing a setting in the PC's BIOS through the BIOS Setup Utility, opened by pressing F1 during startup.

Files installed include:

File	Description
isserver.exe	In-Sight server program
In-Sight.chm	HtmlHelp file
hhupd.exe	Updates IE 4.0 to run compiled HtmlHelp Files
readme.txt	Up-to-date information supplementing the standard documentation.

═══════════════════════════ COGNEX In-Sight

◄ ▲ ► Install: Testing the Installation

You test the installation by starting In-Sight and viewing an image.

▶ **To Test the Installation**

1. Switch on the VGA monitor.
2. Apply power to the In-Sight processor.
3. Look for the In-Sight splash screen.
4. On the Control Pad, click △ In-Sight opens the System menu.
5. From the System menu, click Down to highlight the **Live** option.
6. Click ✕. In-Sight hides the worksheet and displays a live image.
7. Adjust the camera and lens:
   o If the image is completely black, maybe the camera is improperly connected; or maybe the aperture is closed; or maybe there is insufficient light. Check the light, lens, and cable.
   o If the image is blurry, completely gray, or completely white, adjust the camera's position, focus ring, and aperture ring until a sharp image appears on the display.
8. Press any button to halt live video test.

COGNEX In-Sight

◀ ▲ ▶ Install: Testing the Lights

After In-Sight starts successfully, you can test variable-intensity lights (if present)

▶ **To Test the Lights**

1. ??From the TBD menu, select TBD, and TBD.
2. ??In the AcquireImage function, set the light intensity value to ???. The lights should increase to maximum brightness.
3. Set light mode value to ??. The lights should ??do what??.
   o If the LEDs remain unlit, the lighting cable may be loose. If it is not loose, contact Cognex as described in Customer Support.
   o If the LEDs are lit, but do not change in intensity as you adjust the light mode value, make sure the lighting cable is connected to the correct lighting port.

COGNEX In-Sight

◀ ▲ ▶ Install: Updating Firmware

The In-Sight system includes built-in software for its user interface, vision processing functions, and so on. If it becomes necessary to update this software, Cognex will issue a revised version that you can transfer to the Vision Processor. This section describes the procedure for updating the firmware.

▶ **To Update the In-Sight Firmware**

1. TBD

2. TBD
3. TBD.

▶ **To Change or Update the Help Language**

1. TBD
2. TBD
3. TBD.

═══════════════════════════════════ **COGNEX** In-Sight

◀ ▲ ▶ Install: Start up and Shut Down

In-Sight starts up as soon as you apply power and runs continuously as long as power is present. It does not have a power switch. You can reset the hardware by intterupting the power. All unsaved data is lost at power down.

▶ **To Start Up In-Sight**

1. Turn on the power. In-Sight starts up, ?displays its splash screen,? and immediately starts executing ??which job??. Goes online immediately if so configured.

▶ **To Shut Down In-Sight**

??No special shut down procedure, other than saving your work??

1. Save any changes in the current jobs.
2. Turn OFF the power.

═══════════════════════════════════ **COGNEX** In-Sight

◀ ▲ ▶ Install: Troubleshooting

Symptom	Possible Cause	Possible Solution
Power LED Dark	1. No power. 2. Polarity reversed. 3. Blown fuse.	1. Confirm that 24v DC is present. 2. Check the wiring. Reversed polarity should trip the fuse. Interrupting the power to correct the wiring should reset the fuse. 3. Interrupt the power. The fuse resets automatically.

Image black, power LED lit	1. Camera or monitor misconnected. 2. No light on subject. 3. Aperture closed.	1. Double-check the connections. In particular, make sure you haven't reversed the serial and camera cables, which both use RJ-45 connectors 2. Confirm illumination. 3. Open the lens aperture ring.
Image gray, white, or blurry	Poor focus	Focus the lens.
I/O lines fail	Misconnected or disconnected	Confirm the electrical continuity.
In-Sight server issues "can't open port" error.	Serial port disabled	Enable the port on the PC. This usually requires changing a setting in the PC's BIOS through the BIOS Setup Utility, opened by pressing F1 during startup.
Online documentation won't open	Wrong browser version	• HtmlHelp requires Microsoft Internet Explorer 4.0 or higher. If you're running an older version, you must upgrade. • Some versions of IE 4.0 require additional software components to recognize a compiled HtmlHelp file. Try running the updater (hhupd.exe), supplied on the In-Sight CD-ROM.

COGNEX In-Sight

◄ ▲ ► Getting Started with In-Sight

This section gives you a hands-on introduction to the In-Sight system. Using an image of an industrial part, it walks you through a sequence of steps common in real-world applications. To get the most benefit from this exercise, you should follow along on your own system, giving you first-hand experience with the Control Pad and worksheet.

The goal is to measure the distance between a curved edge and a straight edge. This measurement depends on finding both edges, even if the part shifts around from one image to the next. The underlying edge-detection steps are straightforeward when considered in isolation. Compensating for differences in location and orientation—crucial for making sure an application is reliable—adds some preliminary steps. The overall flow is:

1. Acquire an Image of the Part
2. Fixture the Part
3. Locate Featues: Straight Edge
4. Locate Features: Curved Edge
5. Measure the Distance
6. Save the Job.

When you're done, you'll have a major part of an inspection application—*without writing a single line of code.*

Don't worry if this section mentions topics without completely explaining them. The idea is to capture the flavor of using In-Sight, not to exhaustively explore it. When done, you should go on to read *Using the In-Sight Interface*, which details how to use In-Sight's GUI. You then should be ready to start developing your own vision applications, referring to this online book for more information whenever you need it.

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ COGNEX In-Sight

◄ ▲ ► Getting Started: Acquire an Image

In this section, you acquire an image that serves as the starting point for the rest of the exercise.

► **To Acquire an Image of the Part**

1. Before you can follow this exercise, first install and start the In-Sight system. On startup, In-Sight displays a worksheet, where you define formulas for vision processing. Through a semi-transparent overlay, it simultaneously displays the acquired image. This feature, called *worksheet overlay*, eliminates the need to alternate between the image and worksheet—a major convenience when setting up a worksheet based on the features in an image. The amount of transparency is adjustable.

2. Display the sample image (a generalized industrial part), print it, and return to this topic.

3. Put the printed copy under the camera. With a 16 mm lens, the working distance is about one foot.
4. You control In-Sight with an 8-button Control Pad. Start by clicking its △button. In-Sight opens the System menu, used to select high-level options:
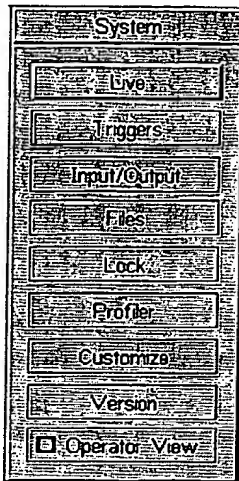
System	
Live	Toggle live mode
Triggers	Continuous, External, or Manual
Input/Output	Serial and Discrete I/O
Files	Save or load jobs or images
Lock	Password locking
Profiler	Analyze performance
Customize	System options
Version	Software information
Operator View	Toggle customized interface

The ✕ and ○ buttons also open important menus, used to build formulas and edit the worksheet.
5. To interactively adjust lens focus and aperture, cursor Up or Down in the System menu and select **Live.** Click ✕, placing In-Sight in live mode, where it passes a stream of images through to the display, a bit like a motion picture camera.
6. Adjust the camera and lens to obtain a clear, focused image with the part oriented horizontally, as shown below.

Image structure, stores acquired image

Current cell

Worksheet rows

Worksheet columns

Formula bar

Worksheet drawn on image
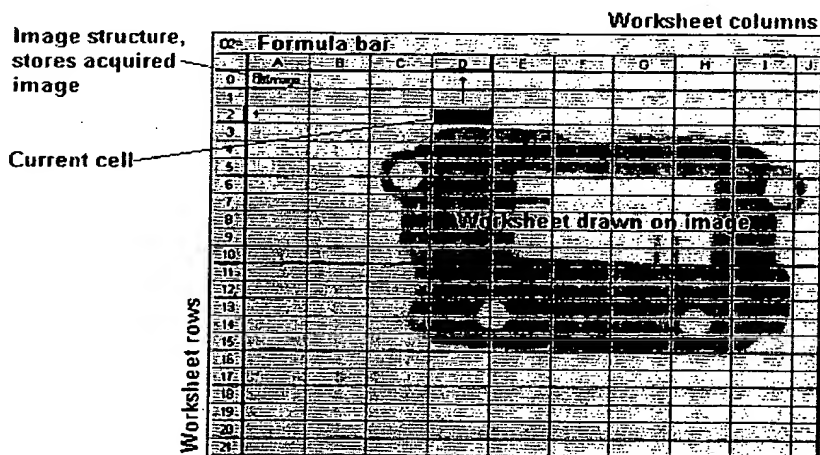
7. When satisfied with the image, click △ to return to the System menu. Click △ again to return to the worksheet. In general, the △ button accepts changes and returns to the previous level. When you exit live mode, In-Sight displays the last acquired image.

Appendix I - Page **33**

You can snap a new image at any time by holding down the ☐ button and simultaneously clicking ✕— a useful shortcut during development. A typical deployed application instead acquires an image when it receives an external trigger signal. In-Sight consequently offers configurable serial and discrete I/O interfaces for connecting external trigger sources and other devices.

8. <u>Continue</u> to the next topic, where you build a model subsequently used to search for two holes.

═══════════════════════════════ **COGNEX In-Sight**

◄ ▲ ► Getting Started: Fixture the Part

Manufacturing processes usually do not perfectly constrain the parts being inspected. To compensate for image-to-image variation in part location and orientation, many vision applications establish a *fixture coordinate system*, which remains fixed with respect to each part, eliminating the image-to-image differences. To do so, the application first finds a "landmark" feature in each image. As much as practical, this feature must be something that the application can find reliably in all situations it is likely to encounter. Subsequent operations are defined by offsets from the fixture origin and depend on its reliablity.
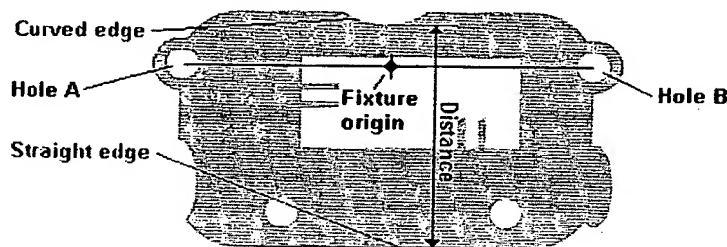
A common strategy is to find the fixture origin and angle through a preliminary <u>feature extraction</u> step. In this exercise, you use <u>pattern matching</u> to find the fixture origin and angle. The basic idea is to find two holes, calculate the midpoint and angle of the line connecting their centers, and adopt those values as a fixture. The steps are:

1. <u>Build a Model</u>
2. <u>Find Two Holes</u>
3. <u>Calculate the Origin</u>
4. <u>Calculate the Angle.</u>

These sections introduce many fundamental In-Sight concepts.

═══════════════════════════════ **COGNEX In-Sight**

◄ ▲ ► Getting Started: Build a Model

Pattern matching is a model-based search technique. It compares a *model*—an example of the desired feature—to regions of an image to find regions that fit the model. In this topic, you extract a model from the image. Along the way, you learn some important In-Sight techniques.

► **To Build a Model of a Hole**

1. Move to cell A1.
2. Click and release the ✕ button, opening the Formula Builder dialog, used to create and edit formulas:
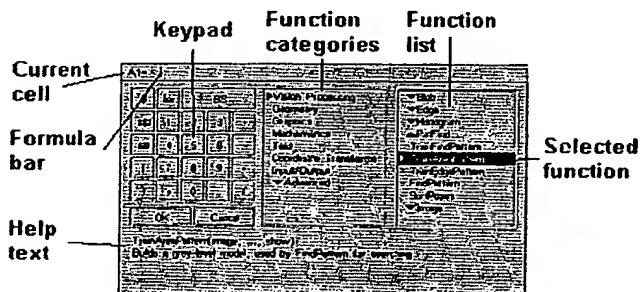
3. In the center pane, highlight **Vision Processing** and click Right, moving to the right pane.
4. Click Down to highlight the **Find** heading, and then click ✕ to expand it.
5. Select the FindPatterns() function (which extracts a model from an image) and click ✕, opening its property sheet:
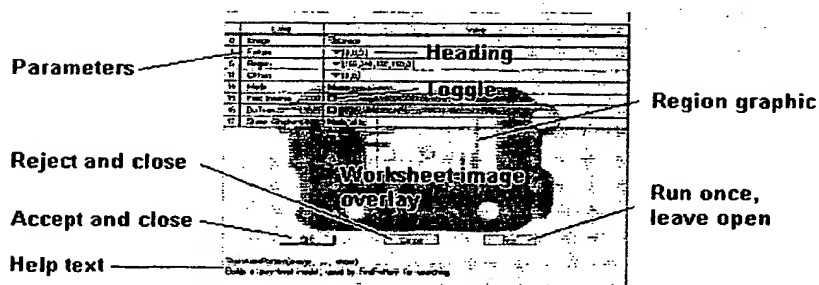
Many In-Sight functions have property sheets, used to view and change their parameter values.
6. Click Down to highlight the Region heading, which lists the parameters that define the region of interest (ROI) used as a model. By default, the Region heading is collapsed (▼). To view the individual parameters, expand the heading by highlighting it, clicking and holding ✕, and selecting Expand from the Enter menu. The heading opens up when you release the ✕ button:

Heading ——— | 5 | Region | | ▲ (296, 237, 150, 150, 0) | ——— Summary
Parameters ———

6	X		296.000
7	Y		237.000
8	High		150.000
9	Wide		150.000
10	Angle		0.000

——— Current values

7. To set the ROI interactively, highlight any Region parameter, and then click ✕ In the resulting dialog box, click the Region button. In-Sight hides the worksheet and draws an interactive cursor on the image. Repeatedly clicking ○ cycles through the cursor modes. One mode sets the region's postion, another its size, and another its angle:

Position   **Size**   **Angle**

Col   Wide   Degrees CCW

Row   High
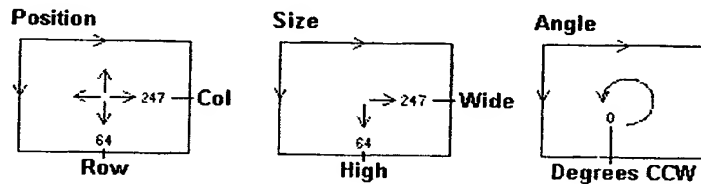
The arrows on the "top" and "left" sides point away from the origin, for operations affected by the region's orientation.

8. Using the Control Pad cursor, set the ROI position by dragging the Region cursor until the center of the ROI is at the center of Hole A. Next, click ○ to change the Region cursor to its resize mode. Resize the ROI to surround Hole A, clicking Up and Down to set the height and Right and Left to set the width. Exclude the background, like this:
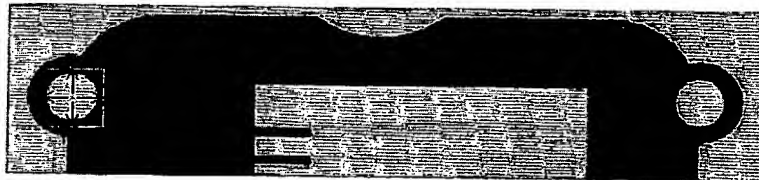
9. Click ✕ to accept the ROI. In-Sight returns to the FindPatterns() property sheet. This kind of interactive, graphical selection for defining regions and other parameter values is a common convenience in the In-Sight interface.

10. Click **OK.** FindPatterns() extracts a model from the ROI and stores it in cell **A1** in a Patterns structure. In-Sight closes the FindPatterns() property sheet and returns to the worksheet, marking the structure with a ⛁ icon. Encapsulating complex image or feature data in a single cell by storing it in a structure is another characteristic feature of the In-Sight interface.

11. Now that you've extracted a model, you're can use it to find Holes A and B.

COGNEX In·Sight

◄ ▲ ► Getting Started: Find Two Holes

In the previous topic, you extracted a model from the image. Now you're ready to search for the model in the image, using it to find Holes A and B.

► **To Search for Holes A and B**

1. With cell **A1** still selected, note the graphic drawn on the image to indicate the model region. Now move to cell **A2**, immediately below the cell storing the Patterns structure. Because the relevant structure is no longer selected, the region graphic vanishes. Displaying a graphic appropriate to the selected structure or parameter is called *flyover graphics*. When you traverse ("fly over") the worksheet, you therefore see a context-sensitive indication of the underlying data.
2. Leaving A2 empty, move to **A3**. Click ✕ to open the Formula Builder, move to the left pane, and open **Find** heading.
3. Highlight the FindPatterns() function (which finds instances of a model in an image) and click ✕ to open its property sheet.
4. To specify the model built previously, you make a reference to the Patterns structure that stores it. To do so, highlight the **Model** parameter and click ✕. In-Sight hides the property sheet and shows the worksheet in *interactive reference mode*. As its name suggests, this view of the worksheet grid lets you define a cell reference graphically, through the Control Pad cursor.
5. Click Up to move the highlighted cell, indicated by the marquee, to cell **A1**, which contains the Patterns structure:

Image object (stores acquired image)
Pattern object (stores model data)
Reference from A2 to A1

	$A\$1 = TrainAreaPattern($A\$0,0,0,0,182,112.5,36,35,0,0,0,0,0,0,0)						
	A	B	C	D	E	F	G
0	⓪Image						
1	⑤Pattern		Marquee indicates				
2	⑥Pattern		target of reference				

6. Click ✕ to accept the reference. Any cell reference can be absolute (unchanged when copied) or relative (keeps a fixed offset, but not a fixed target, when copied). By default, this particular reference is absolute, so a copy of cell A2 uses the same model, stored in cell **A1**.
7. Specify the search region by highlighting the Region heading and clicking ✕. The interactive technique for setting the search ROI is the same as that for selecting the model ROI:
   a. Drag the cursor center to the ROI center.
   b. Click ○ to change the cursor to its two-arrowed form, then resize to surround Holes A and B and exclude Holes C and D. In general, the size of an ROI depends on positional and angular uncertainty and other factors. For this exercise, allow some "padding" on all sides of the holes.
   c. Click ✕ when done to accept the ROI and return to the property sheet.

The result should look about like this:

Region cursor

Region center
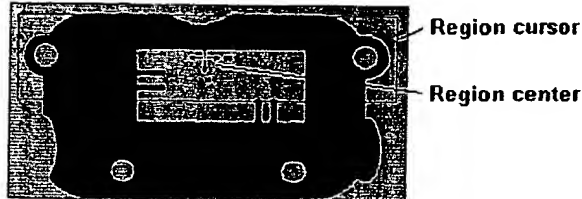
8. Highlight the **Number to Find** parameter and click ✕ to select it for editing. When you select a numeric value, cursor Up increments the value, Down decrements it, and Right and Left move "back and forth" from digit to digit. Increment **Number to Find** to 2, reflecting the number of holes expected in the ROI:

| 17 | Number To Find | 2 |

9. Similarly decrease the **Accept** value to **20**. An <u>acceptance threshold</u> is a the minimum score accepted as a valid instance. Reducing it makes the search more permissive, decreasing the risk of failing to find both holes during this exercise. After setting **Accept**, click ✕

10. In the property sheet, click **OK**, running the model. FindPatterns() returns a Patterns structure, which contains a data about the instances of the model found in the image, to cell A2:

Poses object (stores instances found in region)

Indexes of the two instances found

Search region

Instance 0

	A	B	C	D	E	F	G	H	I
0	@Image								Labels
1	@Pattern	Index	Row	Col	Angle	Scale	Score	(if row
2	@Poses	0.000	188.041	121.027	0.000	100.000	100.000	is empty)
3		1.000	187.691	492.605	0.000	100.000	96.500	
4								Emitted
5								formulas
6						Instance 1		
7								
8								
9								

Each instance has an index and a set of measured values. Because the Patterns structure contains complex feature data, you can't access the data directly from a formula. To expose the individual values, In-Sight automatically emits formulas to empty cells. Other formulas can then refer to these cells to obtain the individual values.

11. At this point, you've obtained all the data you need to calculate the midpoint between the centers of the two holes, as described in the <u>next topic</u>.

═══════════════════════════════════ **COGNEX** In-Sight

◄ ▲ ► Getting Started: Calculate the Origin

In the <u>previous topic</u>, you found two holes through a pattern match search. The result was a Patterns

structure storing the instances found and a set of automatically generated cells containing data about them. In this topic, you use worksheet functions to calculate the midpoint, which will define the origin of the fixture coordinate system.

▶ **To Calculate the Fixture Origin**

1. Except for its automatic generation, an emitted formula is in every respect the same as any other formula. You can freely copy, edit, delete, or otherwise change any emitted formula. For this exercise, the only values you need are the row and column coordinates of the two center points. To simplify the worksheet, delete the formulas that emit the Angle, Scale, and Score values:

   a. Move to cell **E1**, which contains the string "Angle" used as a column label.
   b. Click and hold the ○ button, then click cursor Down and cursor Right to highlight the range from **E1** to **G3**:

1	⬡Pattern	Index	Row	Col	Angle	Scale	Score
2	⬡Poses	0.000	201.232	133.566	0.000	100.000	98.494
3		1.000	204.470	503.648	0.000	100.000	88.689

   c. Release ○. In-Sight opens the Edit menu:

Edit	
Copy	Copy cells to clipboard
Cut	Cut cells, copy to clipboard
Paste	Paste cells from clipboard
Clear	Remove cells, no copy to clipboard
Expression	Open Formula Builder dialog
Worksheet	Customize the worksheet
Cell State	Enable or disable cells
Hide Rows	Hide selected rows
Unhide Rows	Unhide hidden rows
Insert Rows	Insert new row above
Remove Rows	Delete current row, shift up
Col Width [0]	Set column width or row height

   d. Select **Clear** and click ✕. In-Sight removes the formulas from the selected cells.

2. Next, calculate the row coordinate of the midpoint. The formula required is **(C2 + C3)/2**, which you create as follows:

   a. Move to cell **C4** and click ✕ to open the Formula Builder.
   b. Click cursor Right to enter the keypad area. There, move to the (button and click ✕. In-Sight

Appendix I - Page 39

adds the (character to the formula line.

c. Move to the ▦ (relative reference) button and click ✕. In-Sight hides the Formula Builder and shows the worksheet in interactive reference mode. Move the marquee to cell **C2**, which contains the row coordinate for Hole A, and click✕ to accept the reference.

d. Continue in the same fashion to construct the entire formula, **(C2 + C3)/2**.

e. When done, click **OK**. The formula returns its result—the average of the row values—to cell **C4**.

3. After calculating the row average, it's easy to calculate the column average:

a. Highlight cell **C4**, the row average.

b. Click O, yielding the Edit menu.

c. Select **Copy** and click ✕. In-Sight copies the formula to the clipboard.

d. Highlight cell **D4**.

e. Click O, select **Paste** from the Edit menu, and click ✕. In-Sight pastes the formula from the clipboard to cell **D4**:

1	⊕Pattern	Index	Row	Col
2	⊕Poses	–0.000	201.232	133.566
3		1.000	204.470	503.648
4			202.851	318.607

Because the references to the row values are both relative references, In-Sight adjusts them to preserve equivalent offsets. Without any editing, the new formula in D4 is **(D2 + D3)/2**, which calculates the column average. The ability to copy a formula to a new cell and get a similar result is a major benefit of using a worksheet interface for vision applications.

4. Now that you've found the row and column averages, go on to the next topic to calculate the angle.

─────────────────────────────────────── **COGNEX In-Sight**

◄ ▲ ► Getting Started: Calculate the Angle

In the previous topic, you calculated a row and column value to serve as the origin of the fixture coordinate system. Here, you calculate the fixture angle, often called *Theta* to distinguish it from other angle values.

► **To Calculate the Fixture Angle**

1. Highlight cell **E4** and click ✕to open the Formula Builder.

2. From the Geometry category, open the Measure heading and select the **PointToPointAngle** function, which returns the angle between the Y axis and the line defined by two points. Click✕ to open its property sheet.

3. Highlight the **Point 0** heading and click✕, placing the worksheet in interactive reference mode.

4. Move the marquee to cell **C2**, click and hold O, and click cursor Right to select both **C2** and **D2**, the row and column values for the center point of Hole A.

5. Click ✕. In-Sight defines Point 0 through the references to cells **C2** and **C3** and returns to the property sheet.
6. Repeat the process for Point 1: Highlight **Point 1**, click ✕, move the marquee to cell **D2**, select the range **D2:D3** and click ✕. In-Sight defines the second point and returns to the property sheet.
7. Click **OK**, closing the property sheet. Click **OK** again, closing the Formula Builder. PointToPointAngle returns the angle value to cell **E4**:

PointToPointAngle(C2,D2,C3,D3)					
	A	B	C	D	E
0	Image				
1	Pattern	Index	Row	Col	
2	Poses	0.000	201.232	133.566	
3		1.000	204.470	503.648	
4			202.851	318.607	89.499
5					

        Fixture X   **Fixture Y**   **Fixture Angle**

8. Vision applications use a fixture coordinate system to eliminate image-to-image variation in part location and orientation. To see this in action, try the following experiment:
   a. Click and release △ to open the System menu.
   b. Select Triggers and click ✕, opening the Triggers dialog:

    From Control Pad
    Automatic and continuous
    From hardware trigger
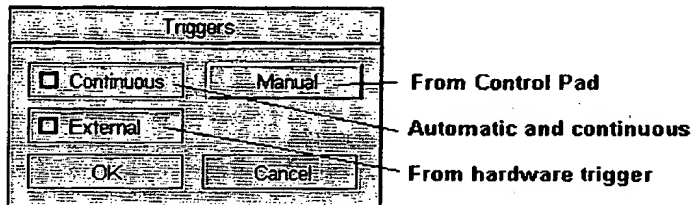
   c. Highlight **Continuous** and click ✕ This enables *continuous acquire*, which constantly updates the image and the worksheet, a bit like a motion picture camera. You should see the worksheet values continuously updating.
   d. Tap the printed image under the camera. The calculated fixture values should follow the image of the part.
   e. Assuming that small disruptions don't cause a problem, try moving the image slightly. The amount of tolerance to change in location and angle depends on the exact size of your region of interest and other variations beyond the control of this exercise.
   f. If you like, you can complete the rest of the exercise with continuous acquisition enabled (⊠). The discussion assumes that disable it (☐) by selecting Continuous, clicking ✕, and then clicking △ twice to return to the property sheet.
9. Now that you've obtained a coordinate system that is constant within the part instead of the image, you're ready to find a straight edge.

COGNEX In-Sight

◄ ▲ ► Getting Started: Find a Straight Edge

In the previous topics, you used a pattern matching and some calculations to establish a fixture coordinate system. At that stage, the goal was to quickly locate the part in the image, not to make a final inspection. In the next group of topics, you find a straight edge and a curved edged, then measure the distance between them.
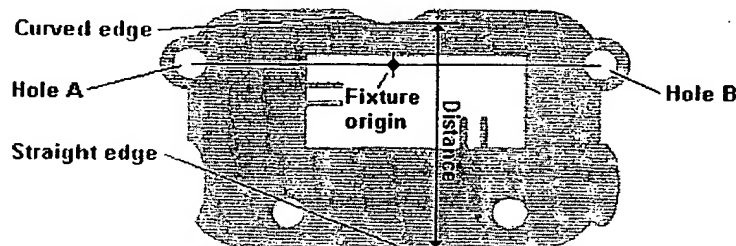
▶ **To Find a Straight Edge**

1. From cell A5, click ✕ to open the Formula Builder.
2. From the Vision Processing category, open the Edge heading, and then open the FindLine property sheet. FindLine, as the name suggests, uses feature extraction method called edge detection to detect straight lines.
3. Highlight the **Fixture** heading and click ✕. In-Sight shows the worksheet in interactive reference mode, so that you can create the references interactively.
4. Highlight cell **C4** (the fixture row), click and hold ○, and then click Right to select cells C4, C5, and C6 (the fixture row, column, and angle).
5. Click ✕ to accept the references, returning to the FindLine property sheet.
6. Highlight the **Region** heading and click ✕. In-Sight displays the Region cursor.
7. An edge is a black-to-white or white-to-black transition. When you define a region for edge detection, you have to consider the edge's *polarity*, which is the expected grayscale transition. In-Sight evaluates polarity along the Region's X axis, progressing away from the origin:
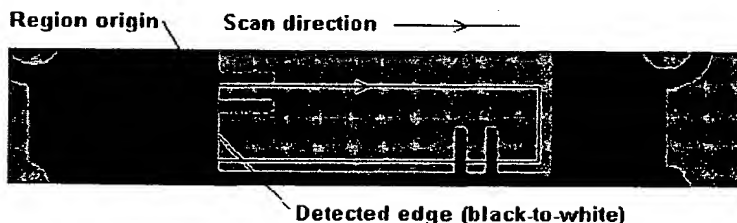
The bottom edge of the part image is horizontal. Viewed from background-to-foreground (bottom-to-top in the image), the edge is white-to-black. To detect this edge, you must rotate the region by 90 degrees, so that its X axis is perpendicular to the edge:

1. With the cursor in its four-arrow form, move the region until its center reaches bottom edge.
2. Click ○ to change the cursor to its two-arrow form, then reduce the width and increase the

height.

3. Click O to change the cursor to its circular form, which rotates the region. Click and hold Left to rotate the region by 90 degrees. The result should look about like this (with the edge strength-graph removed for clarity):
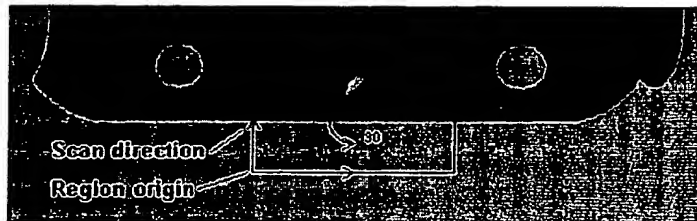
4. Click X to return to the property sheet.

8. Highlight **Polarity** and click X to show its list of values:

10	Polarity	black-to-white
11	Find By	white-to-black
12	Accept Thresh	either

Select "white-to-black" and click X to accept.

9. Click **OK**. FindLine detects the edge and stores an Edges structure. It also emits a row and column for both end points as well as a score:

	A	B	C	D	E	F	
0	Image						
1	Pattern	Index	Row	Col			
2	Poses	0.000	188.636	122.296			
3		1.000	188.931	494.044			
4			188.784	308.170	89.955		
5	Edges	371.703	258.476	371.704	377.476	53.734	

Edges array (stores detected edges)

Endpoint   Endpoint   Score

10. FindLine, like other Edge functions, also draws and edge-strength graph, best viewed by clicking △ to hide the worksheet:
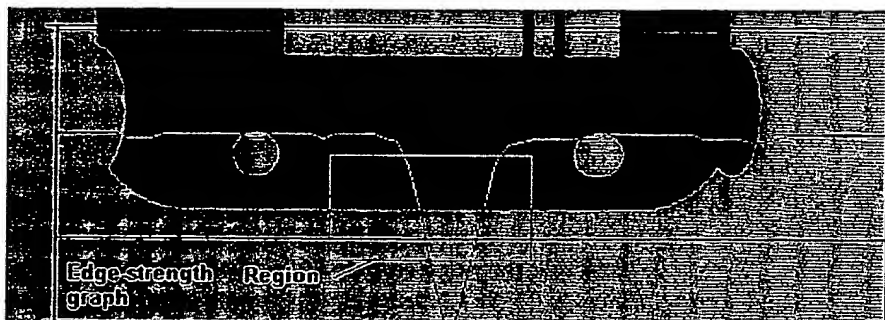
11. Now that you've located a straight relative to the fixture, you're ready to locate a curved edge.

=== **COGNEX** In-Sight

◄ ▲ ► Getting Started: Find a Curved Edge

In the previous topic, you found a straight edge. Here, you use similar techniques to find a curved edge.

▶ **To Find a Curved Edge**

1. From cell **A6**, click ✕ to open the Formula Builder. From Vision Processing, open the Edge, and then open the FindCurve property sheet. FindCurve, as the name suggests, uses edge detection to detect curved lines.
2. As before, highlight **Fixture**, click ✕ to enter interactive reference mode, highlight cell **C4**, click and hold ○, and click Right to select cells **C4**, **C5**, and **C6**. When done, release ○ and then click ✕ to accept the Fixture references, returning to the FindCurve property sheet.
3. Highlight the **Region** heading and click ✕. In-Sight displays the Region cursor. The adjustments are similar to those in the previous step:
    1. With the cursor in its four-arrow form, place the center of the cursor on the desired edge.
    2. Click ○, then reduce the height and width.
    3. Click ○ twice to change the cursor to its curve form. Click and hold Up to warp the region by 100 degrees. The result should look about like this:
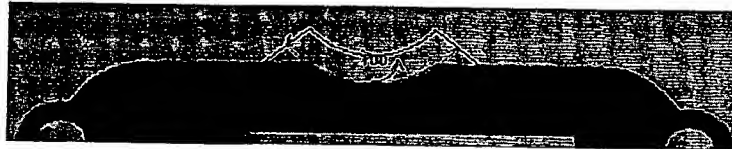
    4. Click ✕ to return to the property sheet.
4. Highlight **Polarity**, click ✕ to show its list of values, select "white-to-black", and click ✕ to accept.
5. Click **OK**. FindCurve detects the curved edge, stores an Edges structure, plots an edge-strength graph, and emits various values:

5	⊟Edges	371.703	253.476	371.704	377.476	53.784			
6	⊟Edges	61.981	303.748	123.737	254.020	122.310	354.584	64.934	37.652
		Endpoint		**Midpoint**		**Endpoint**		**Radius**	**Score**

6. Now that you've located two edges, you're ready to measure the distance between them.

=== **COGNEX** In-Sight

◄ ▲ ► Getting Started: Measure a Distance

At this point, you have found a straight edge and found a curved edge. Now you can measure the distances between them, representing an inspection step in an application.

► **To Measure an Edge-to-Edge Distance**

1. From cell **A7**, click ✕ to open the Formula Builder.
2. From **Geometry**, open the **Measure** heading.
3. Select the **LineToCircle** function and open its property sheet.
4. Select the **Line** parameter, click ✕ to enter interactive reference mode, move to cell **B5**, hold down the ○ button, and select the range **B5:E5** (both end points of the bottom line).
5. Release ○, and then click ✕ to accept the references, returning to the property sheet.
6. ??Select the **Circle** parameter, expand it, select the Row parameter, click ✕ to enter interactive reference mode, move to cell **B6**, and click ✕ to accept, returning to the property sheet.
7. ?Select the Column parameter, click ✕, move to cell **C6**, and click ✕ to accept, returning to the property sheet.
8. ?Select the Radius parameter, click ✕, move to cell **H6**, and click ✕ to accept, returning to the property sheet.
9. Click **OK** to close the property sheet. LineToCircle()
   returns the shortest distance, in pixels, from straight edge to the curved edge:

	A	B	C	D	E	F	G
0	Image						
1	Pattern	Index	Row	Col			
2	Poses	0.000	188.635	122.296			
3		1.000	188.931	494.044			
4			188.784	308.170	89.955		
5	Edges	370.703	258.426	371.704	377.476	53.731	
6	Edges	84.980	303.748	121.737	254.020	122.380	354.504
7		370.703	303.746	146.915	303.747	224.788	
8							
9							
10							
11							
12							
13							
14							
15							
16							

Many inspection applications obtain distances like the one just measured to confirm that a part fits its sp In-Sight includes a wide selection of functions (arithmetic, logic, trigonometry, statistics, range checking, conditionals, and so on) for processing values in formulas.

10. Repeat this sequence of steps to measure the other center-to-center distances.
11. Go on to the next step, to find the leads.

COGNEX In Sight

◄ ▲ ► Getting Started: Save the Job

By now, you're ready to save the job.

Curved edge ⎯
Hole A ⎯
Fixture origin
Distance
Hole B
Straight edge ⎯
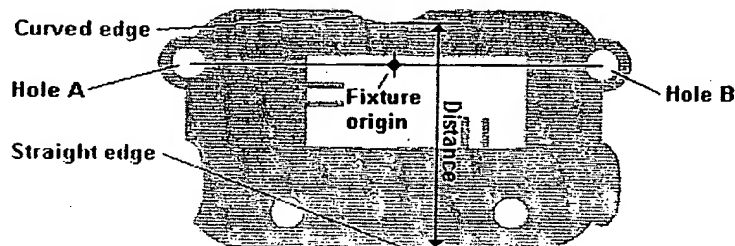
► **To Save the Job**

1. Click and release △, opening the System menu.
2. Select **Files** and click ✕, opening the Files menu:
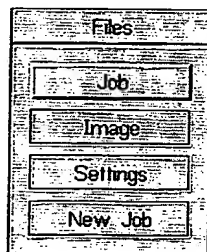
Files	
Job	**Opens Save and Load Job dialog**
Image	**Opens Send and Receive Image dialog**
Settings	**Opens Send and Receive Settings dialog**
New Job	**Create a new job**

3. Select Job, opening the Save and Load Job dialog:

Save & Load Job

Current Job
My Job
Rename   Save →
Delete   Load ←
OK

Slots
► 0 – My Job
1
2
3
4
5
6
7
8

PC
► NO LINK
Send →
Receive ←

Appendix I - Page **46**
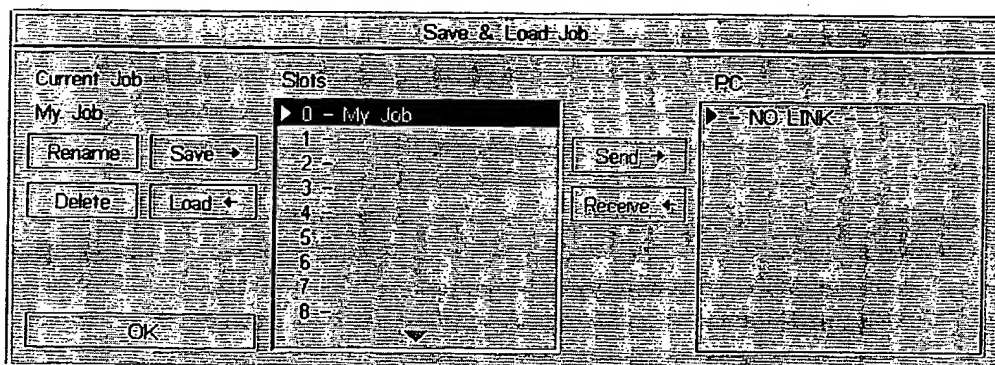
4. Edit the name:
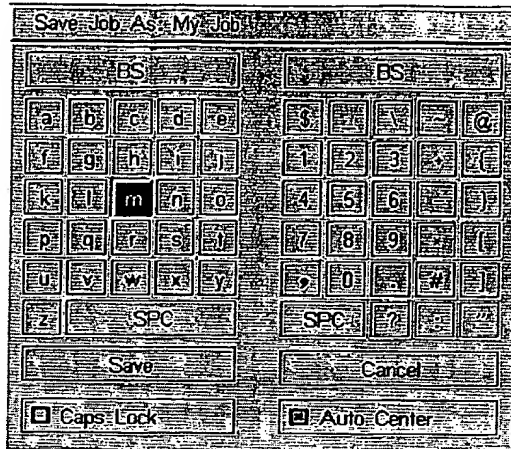
5. Save the job.

You're done.

COGNEX InSight

◄ ▲ ► Control Pad: Working With

You operate and configure In-Sight with an 8-button Control Pad, not with a keyboard or a mouse. This aspect of In-Sight's interface reflects its design for the factory floor, which often does not have a desk or other surface suitable for a keyboard. For In-Sight, the Control Pad is better than a keyboard, since an operator can use it while standing, moving, wearing gloves, and so on. A Control Pad can hang from the ceiling, and its simplicity improves robustness.

This section describes how to use the Control Pad, discussing these topics:

- Button Usage
- Button Menus
- Entering Digits
- Entering Text
- Editing Text

The Control Pad is required for development but optional for deployment. When the system is running as a freestanding device, without the Control Pad, it relies purely on external triggers.

In-Sight automatically calibrates the attached Control Pad on power up. To calibrate it manually, simultaneously press the ✕, ▢, △, and ○ buttons.

To trigger image acquisition from the Control Pad, see *Manually Triggering an Acquire*.

COGNEX In-Sight

◄ ▲ ► Control Pad: Button Usage

The In-Sight Control Pad has a directional cursor and four input buttons. General conventions are:

- The cursor buttons—Up, Down, Right, and Left—are the navigational controls:
  - They provide two-dimensional movement in the worksheet, menus, dialog boxes, and property sheets. For example, Up and Down move along worksheet columns, and Right and Left move along worksheet rows. If a direction is unavailable, its button does nothing.

o When editing a digit, Up increments and Down decrements; and Right and Left move "back and forth" from digit to digit.
o When resizing a region of interest, Up and Down set the height, and Left and Right set the width.
o Holding O, Up selects an enter column, Right selects and entire row, and Down and Right select a cell range.
- The ✕ (Enter) button enters, accepts, and "drills down" hierarchically:
  o Clicked and held on a cell, it opens the Enter menu, used to enter worksheet values, formulas, and references.
  o Clicked and released on a cell, it opens Formula Builder dialog, used to construct expressions. (This shortcut simply accepts the default from the Enter menu.)
  o In a menu, property sheet, or dialog, it selects an option.
- The △ (System or Escape) button exits, cancels, and Open quote "drills up" hierarchically:
  o From a cell, it opens the System menu, used to select high-level options.
  o It closes the current dialog or property sheet, ?canceling? any change in its settings. (In read-only contexts, ✕ and △ are equivalent.)
  o Repeatedly clicking △ eventually returns to the top level.
- The □ (Image) button:
  o From the worksheet or a property sheet, it shows the image by itself, hiding the worksheet or property sheet. Click again to return.
  o To issue a trigger, hold down □ and then click ✕ By default, the trigger acquires a new image, but you can change its behavior.
  o To toggle continuous trigger, hold down □ and then hold down ✕
  o Clicked when when editing a formula or text string, it is a shortcut for backspace.
- The O (Edit) button:
  o Clicked on a cell, it opens the Edit menu, which contains operations that help you manage the worksheet.
  o Clicked and held on a cell, it selects a range of cells in conjunction with the Control Pad cursor.
  o In the Text Entry dialog, toggles between uppercase and lowercase.

=== **COGNEX In Sight**

◄ ▲ ► Control Pad: Button Menus

System		Edit
Live		Copy
Triggers		Cut
Files		Paste
I/O Settings		Clear
Error Handling	Enter	Formula
Lock	Value	Cell State

Appendix I - Page 49

Profiler		Relative		Hide Cols	
Customize		Absolute		Unhide Cols	
Version		Graphics		Insert Rows	
Close		Formula		Remove Rows	
		Properties		Col Width	10
		Close		Close	
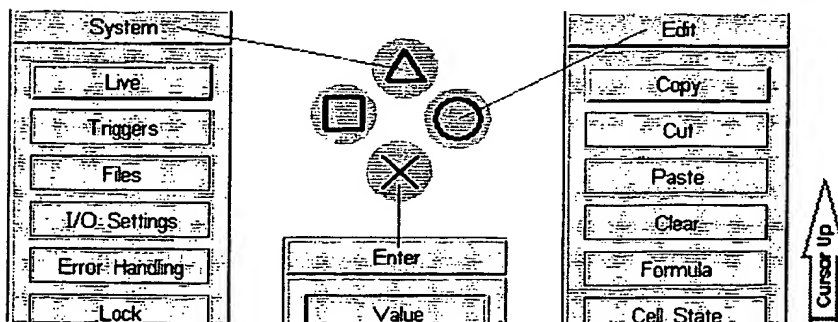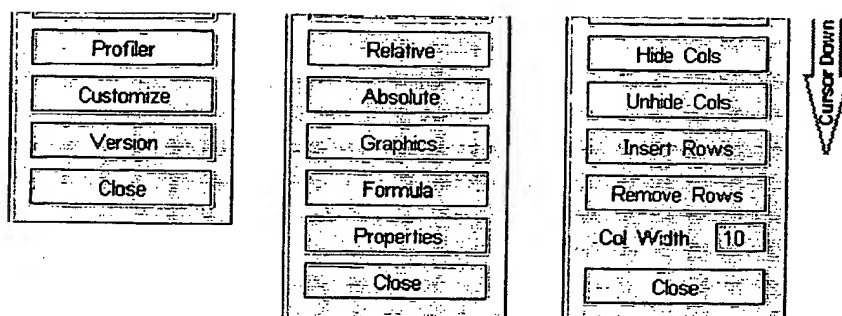
Three of the four Control Pad buttons open important menus. In each menu, In-Sight highlights the current item and highlights (optionally, flashes) its name. Selecting an item and clicking ✕ yields a menu or dialog box where you select or change values through text fields and option buttons. You are always free to switch to any menu to view or change its attributes.

Clicking ✕ opens the current item, letting you view and change its options. For example, clicking ✕ at **Trigger** opens the Trigger dialog. After opening a menu, cursor Up and Down navigate within the menu, and ✕ opens or toggles the selected item. Clicking △ "drills up" one level, ultimately returning to the top level.

COGNEX In·Sight

◄ ▲ ► Control Pad: Entering Digits

You can directly enter numeric values in dialog boxes, property sheets, or worksheet cells using the Control Pad cursor. After selecting a cell, parameter, or option, you increment or decrement each digit with cursor Up or Down, and you move from digit to digit with cursor Right or Left. For worksheet cells, an alternative method is to enter numeric values through the <u>Formula Builder</u>.

► **To Edit a Numeric Value**

1. Select a cell, numeric parameter, or numeric field:
    o From a dialog box or property sheet, highlight a numeric parameter or field, and click ✕.
    o From a worksheet cell already containing a numeric value, click ✕.

    o From an empty worksheet cell, click and hold ✕ to open the **Enter** dialog, select **Value**, and release ✕.

    In-Sight selects the numeric field and highlights the current digit:

Selected digit
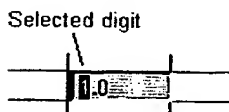
2. To increment the current digit, click cursor Up; to decrement it, click cursor Down.
   NOTE When you increment a digit above 9, it carries over to the next 10. For example, if the value
         is 18 and you select the "8," then clicking Up yields the series 19, 20, and 21. Decrementing
         yields similar behavior, as in the series 21, 20, and 19. Decrementing a positive number
         below 0 yields a negative number, and vice versa.
3. To move from digit to digit—say, from the "10s" space to the "100s" space—click cursor Right or
   Left. In-Sight highlights the selected space:

   [303.9] [303.9] [303.9] [303.9] [303.9]

4. When satisfied with the value, release ✕ to accept it.

―――――――――――――――――――――― COGNEX In-Sight

◄ ▲ ► Control Pad: Entering Text

You enter free-form text in dialog boxes and worksheet cells through the Text Entry dialog, which you use to
build a name.

Current cell   Insertion point   Current string
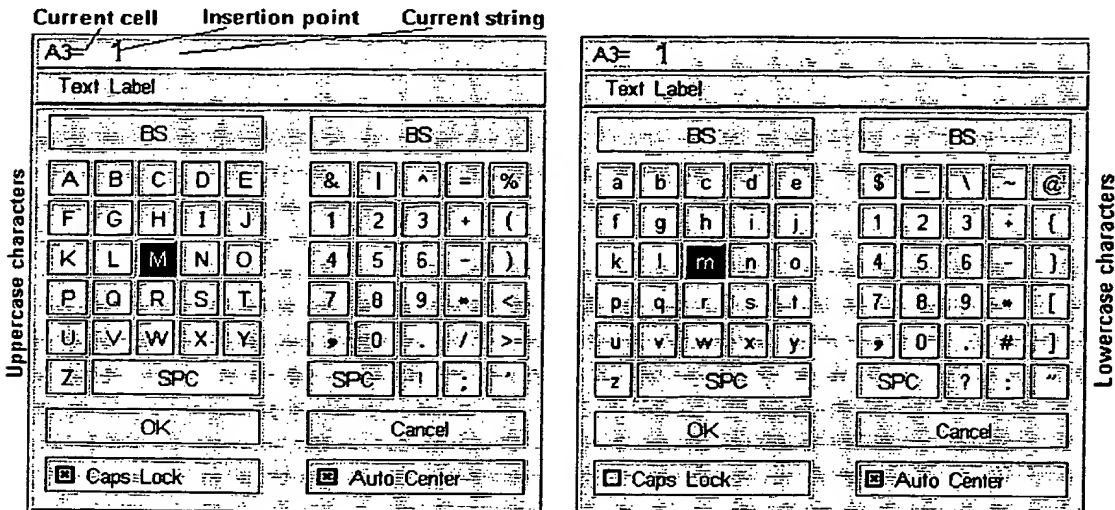
The dialog has two keypad-like areas. On the left, you enter the letters of the alphabet; on the right, you
enter digits, punctuation, and special characters. To toggle between the uppercase and lowercase letters and
characters, click O.

At the top of the dialog, you can see the string under construction. The insertion point, I , marks the current
position for insertion or deletion.

Appendix I - Page **51**

The dialog has the following options:

Caps Lock   Uppercase alphabet and special characters when "on"; lowercase when "off." Also shifts the set special characters. **Shortcut:** click the O button on the Control Pad.

Auto Center   When "on," the cursor jumps to the center position (M or 5) after each character. When "off," the cursor stays at the last position after each character.

The **Caps Lock** option (or its O-button shortcut) toggles between the uppercase and lowercase alphabet; it also changes the set of special characters

Selecting characters from a two-dimensional table for cursor selection is faster than rolling through the entire alphabet for every field. To encourage the use of motor memory, the cursor by default jumps back to the middle position (M or 5) whenever you click OK or reach the perimeter. The click path to each letter is therefore constant. No letter (except Z) is ever more than two clicks away (using diagonals) or four clicks away (ignoring diagonals). This text selection method rapidly becomes second nature.

NOTE   The Text Entry dialog builds names, not worksheet expressions. Its mathematical symbols are text, not operators. For example, the sequence "1+1" is a string, not an expression that evaluates to "2."

COGNEX In-Sight

◄ ▲ ► Control Pad: Editing Text

This topic describes how to edit a text string in the text entry dialog.

► **To Edit a Text String**

1. From a field that accepts text, click X to open the Text Entry dialog. (Or, from the Formula Builder, click [ABC].)
2. Using the Control Pad cursor, move to a letter, digit, or character.
3. Click X. In-Sight adds the letter to the string displayed at the top of the dialog. By default, the text cursor jumps back to the center position, either M or 5.
4. To insert a character inside an existing string, click cursor Up until you reach the string. Next, click Right or Left to move the insertion within the string. With the insertion point in the desired location, click Down, move to the character to insert, and click X. In-Sight adds the new character at the insertion point, between the previous characters.
5. To delete a character, click Up until you reach the string. Next, click Right or Left to move the insertion point in front of the character to delete. Click the ☐ button. (Alternatively, click Down to enter the keypad, then click [BS].) In-Sight deletes the preceding character.

Insertion point

Before | Save Job As: MyZ|String

After | Save Job As: My|String
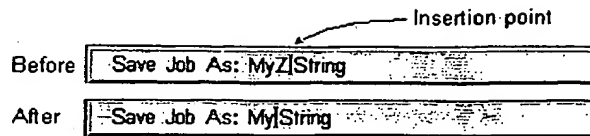
6. When done with the string, click △. In-Sight pastes the string into the field and closes the Text Entry dialog.

COGNEX In-Sight

◄ ▲ ► GUI: Using the In-Sight Interface

You run In-Sight using a graphical user interface (GUI) similar to the interfaces familiar from personal computers. The In-Sight GUI offers menus, dialogs, property sheets, option buttons, text fields, and other familiar devices. The most important part of the In-Sight GUI the *worksheet*, a rectangular array of cells arranged by row and column. A key difference between In-Sight and a personal computer is its use of a Control Pad in lieu of a mouse and keyboard, reflecting practical requirements on the factory floor.

This section introduces you to In-Sight's user interface, discussing its major components:

- Worksheet
- Formula Builder
- Formula Builder Buttons
- Property Sheet
- Using a Property Sheet
- Flyover Graphics
- Understanding Structures
- Table of Structures
- Emitted Formulas.

The In-Sight interface displays the worksheet and image at the same time by a semi-transparent overlay. Viewing the image and the worksheet together eliminates the need to alternate between the two views—a major convenience when setting up a worksheet based on the features in an image.

When done with this section, you should be ready to go to *Using the Worksheet*, which describes how to manipulate cells, rows, and columns.

COGNEX In-Sight

◄ ▲ ► GUI: Worksheet

The most important part of the In-Sight interface is the worksheet. As shown below, a worksheet is a table of cells, with the rows identified by a digit (0 through 999) and the columns identified by a letter (A through Z). You refer to a cell through its row-column address. For example, the top-left cell is **A0**, where In-Sight stores the acquired image.

Image structure, stores acquired image

Current cell

Worksheet columns

Formula bar
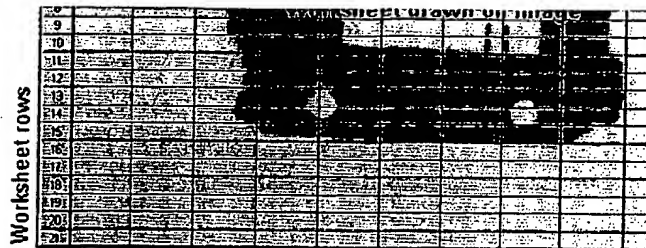
Initially, the worksheet is blank (except for a few cells in row 0, which store the acquired image and other crucial data). Each cell can contain a formula, which is an expression that evaluates to a value. To create a machine vision application, you put formulas into cells by means of the Control Pad.

At a given time, the focus of the worksheet is on one cell. Moving around the worksheet with the Control Pad changes the cell that has focus. In-Sight indicates the current cell by highlighting it in inverse video and by printing its row-column address in the Formula Bar at the top of the worksheet:

Current cell (Row, Column)
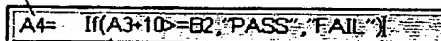
Current formula   Insertion point

The Formula Bar also shows the current formula.

In-Sight displays the worksheet and the image at the same time by a semi-transparent overlay. This feature lets you view the image and the worksheet together, eliminating the need to alternate between the two views—a major convenience when setting up a worksheet based on the features in an image.

COGNEX In-Sight

◄ ▲ ► GUI: Formula Builder Dialog

When you click a cell (or select Formula from either the Enter or Edit menus), In-Sight opens the Formula Builder dialog, which you use to put expressions into worksheet cells:

Help text
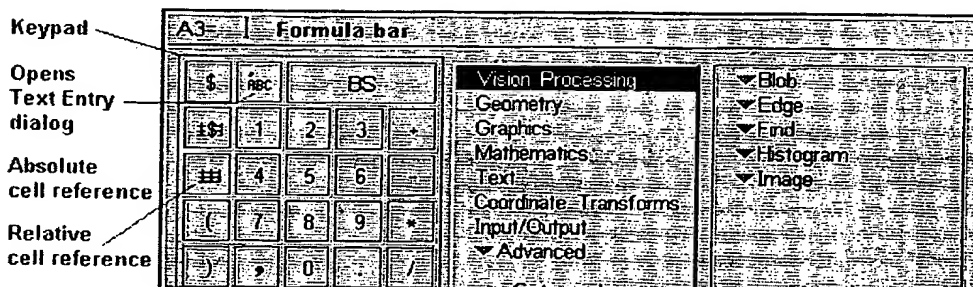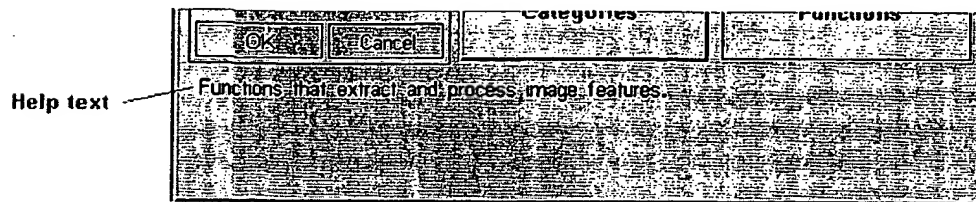
To expand a collapsed heading (marked ▼), select it and click the ✕ button; to collapse an expanded heading (marked ▲), select it an click ✕. In-Sight shows or hides the contents of the heading.

Selecting a function from the function list opens that function's <u>property sheet</u>, though which you set its values. (Some functions don't have property sheets. <u>Pi</u>, for example, doesn't have any parameters, so it doesn't have a property sheet.) For details about using the Formula Builder, see *Entering a Formula*.

COGNEX In Sight

◄ ▲ ► GUI: Formula Builder Buttons

Edits a $ character in a formula, to change a relative reference to an absolute reference.

Opens the Text Entry dialog

Destructive back space

Creates an absolute cell reference. Alternative: select Absolute from the Enter menu.

Creates a relative cell reference. Alternative: select Relative from the Enter menu.

Arithmetical operators.

Parenthesis operators, which control the order of evaluation. All functions require a terminal closing parenthesis.

Decimal point.

Closes the Formula Builder, transferring the formula to the worksheet.

Closes the Formula Builder, erasing the formula.

COGNEX In Sight

◄ ▲ ► GUI: Property Sheet Dialog

You set parameters values through a *property sheet*, opened when you select the function from the <u>Formula Builder</u>.

```
A1 =    ExtractSortBlobs($A$0[,0,0,0,174.5,343,97,230,0,1,-1,0,2,1,100,100000,0)
```

	Label	Value
0	Image	⊞ Image ————— Function name ——— Parameter list
1	Fixture	▼ (0,0,0) ————— Collapsed heading
5	Region	▲ (174.5,343,97,230,0) — Expanded heading
6	X	174.500
7	Y	343.000
8	High	97.000
9	Wide	230.000
10	Angle	0.000
11	Number to Find	1 ——— Parameter list
12	Threshold	-1
13	Include Holes	☑ ——— Enabled toggle
14	Boundary Blobs	☐ ——— Disabled toggle
15	Color: Blob	▶ either List of values
16	Background	▶ white ——— Accepts changes, closes property sheet
17	Area Limit: Min	100.000 ——— Rejects changes, closes property sheet
18	Max	100000.000

[OK]   [Cancel]   [Run]

ExtractSortBlobs(Image, Fixture, Region, ...)   Runs once to test current values,
Extracts blobs from an Image and sorts in one step.   without closing
Returns a Blobs structure.

                                                       Help text

To select the value you want to change, use the Control Pad cursor to move up and down the input list. The flyover graphics and help text both change to reflect the current parameter.

The triangle icons indicate items with more than one parameter. Clicking the triangle opens the item to show all of its parameters.

If a cell contains a function, you can open the property sheet from the cell by moving to the cell, clicking ✕, and selecting Properties from the Enter menu.

COGNEX In·Sight

◀ ▲ ▶ GUI: Using a Property Sheet

The property sheet use icons to indicate types of inputs and relationships among them.

Headings

Headings collect related values.
To expand a collapsed heading (
▼), select it and click the ✕
button; to collapse an expanded
heading (▲), select it an click ✕.

| 5 | Region | ▼ {226.5,199.5,113,243,0} | **Collapsed** |

5	Region	▲ {226.5,199.5,113,243,0}	**Expanded**
6	X	226.500	
7	Y	199.500	
8	High	113.000	
9	Wide	243.000	
10	Angle	0.000	

Toggles

Toggles are input values that
have two states, ON (■) or
OFF (□). To change between
ON and OFF, select the toggle
and click ✕.

| 13 | Include Holes | ■ | **On** |
| 13 | Include Holes | □ | **Off** |

Lists

Lists, marked ▶, are input values
that have several named values.
To expand or contract a list,
click ✕.

| 15 | Color: Blob | ▶ black | **Collapsed** |

15	Color: Blob	▶ black	**Expanded**
		white	
		either	

COGNEX In-Sight

◀ ▲ ▶ GUI: Flyover Graphics

In-Sight shows the worksheet or property sheet and the image simultaneously—a major convenience when
developing a vision application. It also overlays relevant graphics on the image. For example, the following
image shows a property sheet, an image, and graphics indicating the region of interest and the features
extracted from the image:

Image and worksheet
displayed together

— Region graphic

Extracted blobs
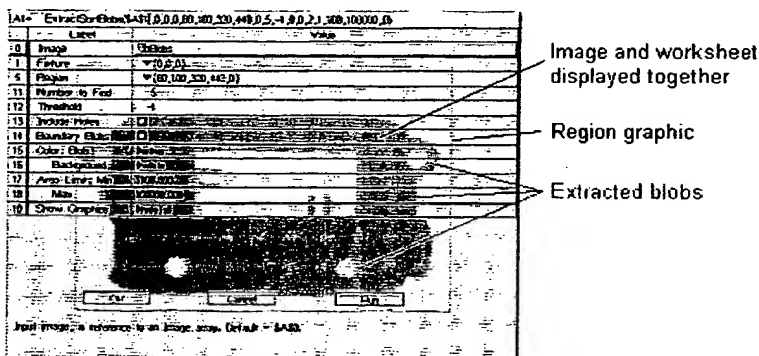
In-Sight always displays the graphics associated with the current cell or property sheet. By default, moving off a cell removes its graphics. When you traverse ("fly over") the worksheet, you therefore see a context-sensitive indication of relevant data. For example, when you highlight a cell containing a Blobs structure, In-Sight draws the blob outlines on the image. When you move away from the Blobs structure, the blob outlines vanish.

If you want the graphics to persist when you move off its cell, you can set the associated function's <u>Show</u> parameter to enable input graphics, result graphics, or both.

NOTE   Sometimes the worksheet grid can hide lines or points that happen to fall exactly under a grid line. The image is the bottom layer, the flyover graphics are the middle layer, and the worksheet is the top layer.

COGNEX In-Sight

◄ ▲ ► GUI: Show Graphics Option

When you highlight a cell containing an <u>structure</u>, In-Sight displays the associated flyover graphics. By default, these graphics vanish when you move to a new cell, reducing visual clutter. If you want them to persist, you can enable them through the associated function's **Show Graphics** parameter, available for functions that return a structure.

For the persistant graphics, In-Sight distinguishes between *input graphics* (which show the function's parameters), *result graphics* (which show a measured value or other result); and *charts* (which plot a histogram, score, or other value). For example, PointToCircle() has input graphics for the point and circle, a result graphic for the distance, and no chart:
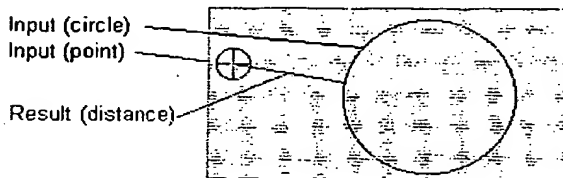
To control the persistant graphics, the Show Graphics parameter offers these options:

hide all	Disables all graphics, which are visible only when you "fly over" the associated cell. The default.
result graphics only	Enables only the result graphics.
input and result graphics	Enables both input and result graphics.
show all	Displays all input, result, and chart graphics.

COGNEX In-Sight

◄ ▲ ► GUI: Understanding Structures

In-Sight stores multi-dimensional image or feature data in *structures*. For example, it stores the acquired image in a structure called Image that resides in cell A0. Functions that depend on the acquired image refer to cell A0 to obtain the image stored there. Functions that produce a new image—for example, as the result of an image processing operation—create a new, additional Image structure to store the result image.

Many vision functions store results in various structures, each marked with an ℬ icon in the worksheet. FindPatterns(), for example extracts a model from an image and stores it in a structure called Patterns. FindPatterns() subsequently refers to the Patterns structure to obtain the model, searches an image for instances of the model, and stores its results in a structure called Patterns. Internally, the Patterns structure is a database of the instances found in the image, each described by a set of values such as its index, center row, center column, angle, scale, score, and so on.

Poses object (stores instances found in region)

Indexes of the two instances found

Search region

Instance 0

	A	B	C	D	E	F	G	H	I
0	ℬImage							Labels	
1	ℬPattern	Index	Row	Col	Angle	Scale	Score	(if row	
2	ℬPoses	0.000	188.041	121.027	0.000	100.000	100.000	is empty)	
3		1.000	187.691	492.606	0.000	100.000	96.500		
4								Emitted	
5								formulas	
6							Instance 1		
7									
8									
9									

A structure always resides by itself in a worksheet cell. You cannot put a structure directly in a formula because a structure is a container for complex data. The mathematical operations used to build formulas don't apply to structures. For example, a formula like "sqrt(Patterns)" doesn't make sense because a Patterns structure does not have a meaningful square root.

To process a value stored in a structure, you must first extract the individual value. To simplify this process, many functions that produce structures automatically emit cells containing individual results. You can then build formulas that refer to these cells or delete any unneeded cells.

For example, suppose you want to average the score values of the first three instances stored in a Patterns structure. Each instance has an index, starting with index 0. You therefore build a formula that refers to the score result for instance 0, 1, and, 2, sums them, and divides the sum by three.

The emitted results are read from the associated structures through Data Access functions. If you prefer, you can call these underlying functions directly.

COGNEX In-Sight

◄ ▲ ► GUI: Structures

Structure	Stores
Blobs	Blobs database. Each blob has an index and a set of measured values.
Calib	Coordinate transformation, generated by Calibrate and subsequently used as a source of calibration data.
Circle	Fixtured circle.
Cross	Fixtured cross mark.
Delay	Buffer, generated by DelayLine.
Dist	Line segment defined by its end points, length, and an angle (from the image row axis, by default).
DNData	Stores DeviceNet data. Created by the DeviceNetRead function.
Edges	Edges detected in an image. Each edge has an index and a set of measured values.
Fixture	Fixture row, column, and angle.
Hist	Histogram and histogram statistics. Generated by ExtractHistogram.
Image	An image. The acquired image is always in cell A0. Other images—say, from NeighborFilter, PixelFilter, or WarpImage—can reside elsewhere.
Line	Fixtured line.
Point	Fixtured point.
Patterns	Stores an area or edge model and the results of a pattern-match search. Each instance found has an index and a set of measured values.
Region	Fixtured region.

COGNEX In-Sight

◄ ▲ ► GUI: Emitted Formulas

Many functions that return an structure automatically emit one or more formulas at creation, eliminating the need to create a new formula by hand for each individual result:

Poses object (stores instances found in region)

Indexes of the two instances found

Search region

Instance 0

	A	B	C	D	E	F	G	H	I
0	ØaImage							Labels	
1	ØaPattern	Index	Row	Col	Angle	Scale	Score	(if row	
2	ØaPoses	0.000	188.041	121.027	0.000	100.000	100.000	is empty)	
3		1.000	187.691	492.605	0.000	100.000	96.500		
4								Emitted	
5								formulas	
6							Instance 1		
7									
8									
9									

Each emitted formula calls a data access function to return one of the values in the structure. Other formulas can refer to the emitted values as data sources. The function inserts its emitted cells in adjacent empty cells, inserting new rows if necessary.

Except for its automatic generation, an emitted formula is in every respect the same as any other formula. You can freely copy, edit, delete, or otherwise change any emitted formula. The reference to the underlying

structure is always an <u>absolute reference</u>. If you copy an emitted formula, the copy therefore keeps the same data source.

Many structures store an indexed list of items. The corresponding emitted formulas occupy a block of cells. Each row is a set of values describing one of the indexed items. Each column is a particular data value—a row, column, angle, scale, or other result.

Some structures store additional values beyond those automatically emitted. The emitted results are simply those thought generally useful. To manually access additional results, you can manually create a new formula with the desired "<u>getter</u>" function.

NOTE In-Sight prints a descriptive label above each emitted value when the worksheet row above the function is unoccupied. If the row already contains a formula, In-Sight omits the descriptions. To see the descriptions, leave an empty row for them. You can edit the descriptions as strings.

COGNEX In-Sight

◄ ▲ ► Modes: Online and Offline

In *online mode*, In-Sight responds to discrete and serial I/O, allowing it to communicate with other equipment in a production application. Online mode therefore represents "normal" operation in a deployed application. Offline mode, in contrast, disables all I/O, isolating In-Sight from other equipment.

You change the mode with a toggle in the System menu. In-Sight indicates the current operating mode in the top-right corner:

○ Online		● Online

► **To Change Operating Modes**

1. Click and release the Δ button, opening the <u>System menu</u>.

2. Select **Online** and then click ✕. In-Sight toggles from online mode to offline mode or vice-versa, indicating the current mode in the top-right corner of the worksheet.

3. Click Δ to close the System menu.

COGNEX In·Sight

◄ ▲ ► Modes: Secure, Operator, Program

A *mode* is ???:
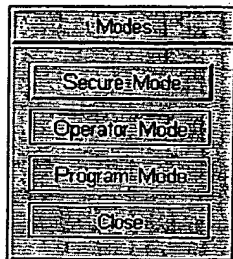
COGNEX In-Sight

◄ ▲ ► Jobs: Working With

A *job* is an executable configuration, like a software program, that contains a <u>worksheet</u>, all of its <u>formulas</u>, and other values. After you set up In-Sight to solve a problem, you can name and save the current job from program memory to non-volatile flash memory, which acts like a solid-state hard drive for for storing jobs. You can subsequently load the job from its slot in flash memory to program memory for execution.

This section describes how to work with jobs, discussing the following topics:

- <u>Saving and Naming a Job</u>
- <u>Loading or Copying a Job</u>
- <u>Creating a New Job</u>
- <u>Selecting the Startup Job</u>
- <u>Archiving Jobs.</u>

NOTE   A job preserves only the worksheet and certain other values. It does not record the current image or the global options for defaults, passwords, I/O, and so on. Settings that affect all jobs are not saved with individual jobs. Instead, they are saved individually whenever changed.

===> **COGNEX In-Sight**

◄ ▲ ► Jobs: Saving, Naming, Deleting

Saving a job writes the current job from program memory to a specified slot in flash memory, preserving any changes applied after the last save. Saving a job captures the current worksheet and all of its expressions, but not the current image or global options such as the password. When you save a job, you can supply a name, subsequently used to load the job.

► **To Save a Job**

1. Click and release the △ button, opening the <u>System menu</u>.
2. Select **Files** and then click ✕, opening the Files menu.
3. Select **Job** and click ✕, opening the **Save and Load Job** dialog.
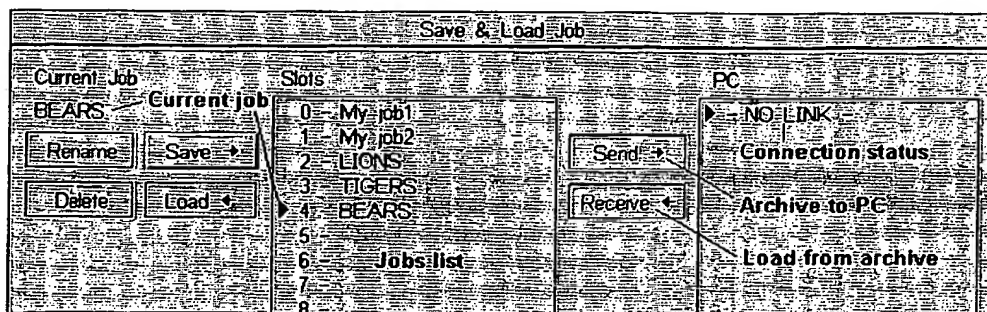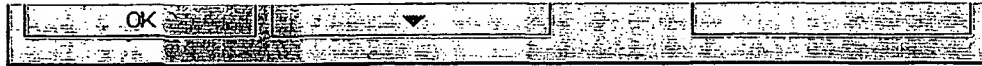
As implied by the dialog's tabular organization, each job slot in flash memory has an index, 0 through 15, and an optional name.

4. In the dialog, highlight a job slot using the Control Pad cursor.
    o To save the job without a name, click △ In-Sight names the job "Unnamed" and saves it to flash memory.
    o To name and save the job, click ✕ In-Sight opens the Text Entry dialog, which you use to build a name as described in Entering Text When done, click **OK**, accepting the name. In the Save Job dialog, click △ to save the named job.

?If you attempt to save a job to a slot that already has a job, In-Sight issues an Are You Sure warning?
NOTE The user-assigned name, up to ?16 characters long, is just a comment, not an alias used in other contexts.

COGNEX In·Sight

◄ ▲ ► Jobs: Loading or Copying

Loading a job copies a named job from flash memory to program memory for execution. ?Currently cannot load multiple jobs?

► **To Load a Job**

1. Click and release the △ button, opening the System menu.
2. Select **Files** and then click ✕, opening the Files menu.
3. Select **Job** and click ✕, opening the **Save and Load Job** dialog.
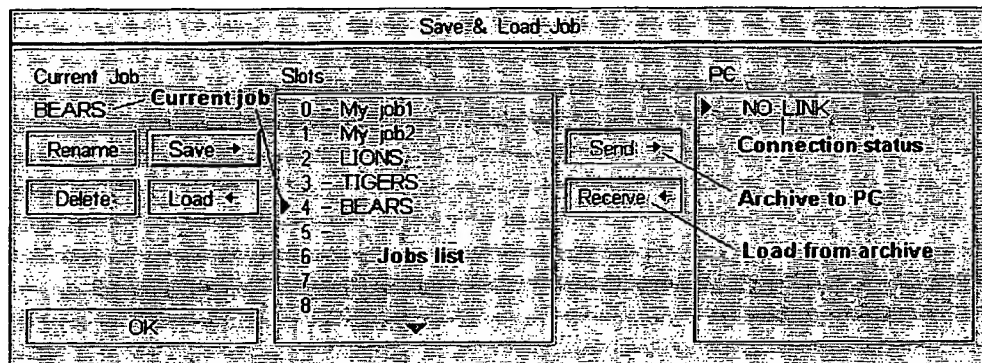
Each job slot in flash memory has an index, 0 through 15, and an optional name.

4. Select the job to load, and then click ✕ In-Sight loads the job from flash memory to program memory. ?Do you get a warning about overwriting an unsaved job?

To duplicate a job, load it from one job slot and save it to a second job slot.

━━━━━━━━━━━━━━━━━━━━━━━━━━► **COGNEX** In-Sight

◄ ▲ ► Jobs: Creating a New Job

Creating a job initializes a new, unnamed job, erasing any existing job after confirmation.

▶ **To Create a New Job**

1. Click and release the △ button, opening the System menu.
2. Select **Files**, opening the Files menu.
3. Select **New Job** and then click✕, opening the New Job dialog:

New Job
Are you sure you want to clear all data from the current job?
OK   Cancel
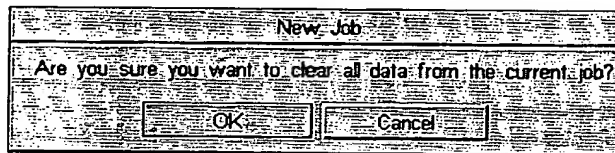
4. Click **OK** to create a new job or **Cancel** to cancel.

Alternatively, click **New Job** in the **Save and Load Job** dialog.

━━━━━━━━━━━━━━━━━━━━━━━━━━► **COGNEX** In-Sight

◄ ▲ ► Jobs: Setting the Startup Job

On power up, In-Sight immediately starts executing the job that was running when it shut down. ??can you specify a default job??

Discuss new autostart dialog

currently, starts up with job active at shutdown??--talk to Nick

You can also change the job over the serial port, as described in ??

COGNEX In-Sight

◄ ▲ ► Jobs: Archiving

To make an archive copy of a job, as a safety archive or for In-Sight–to–In-Sight transfer, the In-Sight can copy ?the entire contents of flash memory? to a hard drive on a remote computer via its RS-232 port.

▶ **To Archive Jobs to a remote PC**

1. ?Hook up the PC to the serial port as described in XXX.
2. ?Start up the archive program.
3. From the XX dialog, specify a name for the new archive file.
4. Click Archive. ?The archive program copies the entire contents of flash memory to the host PC.? ? Could you archive individual jobs?
5. When done, disconnect the PC from the serial port.

▶ **To Archived Jobs from PC to Flash Memory**

1. ?Hook up the PC to the serial port
2. ?Start up the archive program.
3. From the XX dialog, specify the name of the archive file to load.
4. Click Load. The archive program copies ?the entire contents archive file to the.
5. When done, disconnect the PC from the serial port.

To transfer a job from one In-Sight to another, use an archive file

COGNEX In-Sight

◄ ▲ ► Images: Working With

Obtaining an image is a fundamental step for any machine vision application. In-Sight therefore offers several ways to trigger an acquisition.

- Manually Triggering an Acquire (Control Pad)
- Viewing a Live Image
- Enabling a Trigger
- Saving or Loading Images.

You typically use manual or live acquisition during development. You typically used triggered acquisition in the deployed application.

Common image problems include:

- If the image is too light or dark, adjust the aperture ring on the lens. If full aperture yields insufficient brightness or depth of field, increase the light level as described in TBD.
- If the image is blurry or uniformly gray, focus the lens. Alternatively, reposition the camera until you get a sharp image.
- You typically get the best image (and the best lighting response) by placing the camera exactly perpendicular to the subject, particularly if the depth-of-field is shallow.
- Many feature extraction operations depend on a minimum feature size. If the feature of interest is too small, reduce the camera-to-subject distance, increase the focal length of the lens, or both.

COGNEX In-Sight

◄ ▲ ► Images: Manual Trigger

In-Sight acquires a new image and starts running the current job whenever it receives a trigger signal. Several trigger sources are available including the Control Pad, which you typically use during development as a convenient way to grab an image. ??Alternative: use dialog Manual option

► **To Trigger an Acquire from the Control Pad**

1. Hold down the ☐ button.
2. Click the ✕ button. In-Sight immediately acquires an image, for example as shown below:

Image structure, stores acquired image

Current cell

Worksheet columns

Other methods for acquiring an image are live mode, continous acquire, and external triggers, as described in Working with Images.

COGNEX In Sight

◄ ▲ ► Images: Viewing a Live Image

In-Sight offers a *live mode*, which passes the image directly to the display, a bit like a motion picture camera. You typically use live mode when adjusting the lens, since it offers immediate feedback. In live mode, In-Sight suspends the worksheet, updating it only once at exit. With *continuous acquisition*, in contrast, In-Sight constantly updates the image and the worksheet.

▶ **To View a Live Image**

1. Click and release the △ button, opening the System menu.
2. Select **Live**. In-Sight enters live mode, passing the image to the display.
3. If the image is too light or dark adjust the aperture on the lens. If the image is blurry or totally gray, adjust the focus ring. If the subject is outside the field-of-view, adjust the position of the subject and camera until the features of interest are visible in the image.
4. When done, click any button, exiting live mode and returning to the worksheet. At exit, In-Sight acquires the last image and updates the worksheet.

COGNEX In Sight

◄ ▲ ► Images: Enabling a Trigger

An *external trigger* is a hardware signal that causes the camera to acquire an image. In an inspection application, for example, a sensor might detect that a part is in position for inspection an issue a trigger. Most deployed applications rely on external triggers. This section describes how to set up the software to expect an external trigger.

▶ **To Enable an External Trigger**

1. Click and release △, opening the <u>System menu</u>.
2. Select **??** and then click ✕ In-Sight displays the Triggers dialog:

— From Control Pad
— Automatic and continuous
— From hardware trigger
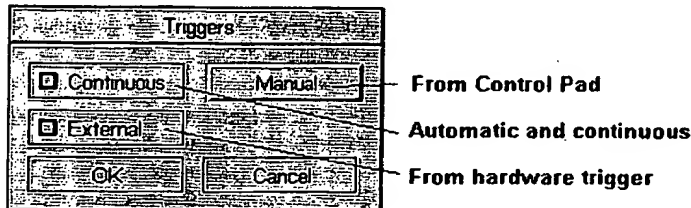
3. XXXX.

???Steve O requests a section showing how to set up an acquire triggered from serial or discrete I/O. It would discusss:

1. Receiving a trigger from serial or discrete I/O
2. Processing the input message
3. As a result of that processing, enable or disable the <u>AcquireImage structure</u> (A0), which results in a HW signal going to the camera and thus an image.

COGNEX In-Sight

◄ ▲ ► Images: Saving or Loading

If you have a serial connection to a remote PC, you can save or load an image.

► **To Save an Image**

1. Click and release △, opening the <u>System menu</u>.
2. Select **Image** and then click ✕, opening the **Send and Receive Image** dialog.
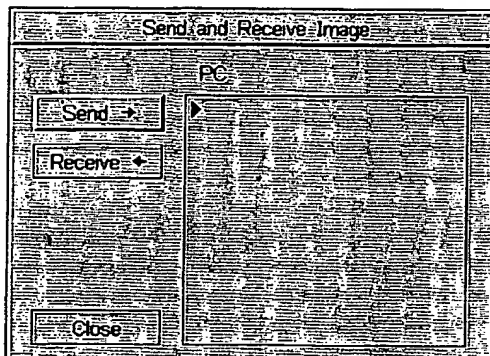
3. Click **Send >**.

▶ **To Load an Image**

1. XXXX
2. XXXX.

?Do we need a separate topic for this guy
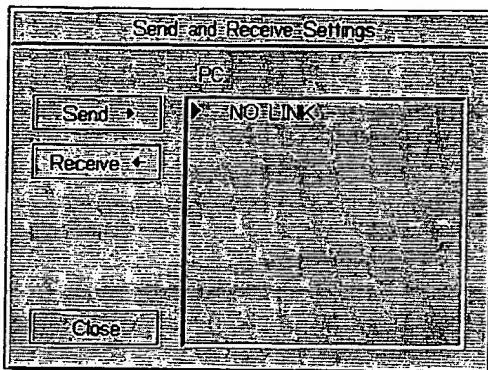
COGNEX In-Sight

◄ ▲ ► Worksheet: Using

In-Sight's worksheet interface greatly simplifies the task of developing and maintaining machine vision applications because it takes advantage of skills already familiar to many developers. With In-Sight, you acquire an image to a worksheet cell and then refer to the cell to apply built-in image processing and feature extraction functions. The results are simple scalar values that you can manipulate with conventional spreadsheet functions. By hiding the complexity of the image data and the algorithms used to manipulate it, In-Sight makes practical vision processing a reality for the manufacturing engineer or technician.

This section describes how to use the worksheet:

- Selecting a Cell or Cell Range
- Selecting a Row or Column
- Cutting, Copying, and Pasting Cells
- Clearing Cells
- Enabling or Disabling a Cell
- Hiding or Unhiding Rows or Columns
- Inserting or Removing a Row
- Setting Row Height or Column Width
- Customizing the Worksheet.

═══════════════════════════════ **COGNEX In-Sight**

◄ ▲ ► Worksheet: Select a Cell or Cell Range

You select a cell or range of cells before performing another worksheet operation, such as cutting or copying cells. To select entire rows and columns, see *Selecting a Row or Column*.

► **To Select a Single Cell**

1. Using the Control Pad cursor, move to the desired cell. In-Sight highlights the current cell and prints its row-and-column address in the Formula Bar:

Current cell
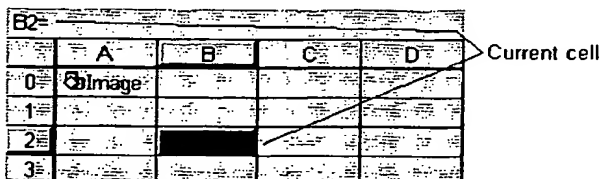
2. Cut, copy, edit, or otherwise change the cell.

► **To Select a Cell Range**

Select entire row

Select
entire
column

Extend
range
right

Extend range down
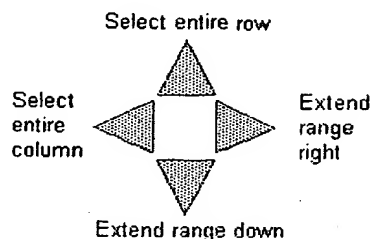
1. Move to the top-left cell in the desired range.
2. Hold down the ○ button.
3. Move to the bottom-right cell in the range by clicking cursor Right or Down to extend the range to the right or down. In-Sight highlights the range:

B2=					
	A	B	C	D	E
0	Image				
1					
2					
3					
4					

4. Release ○. In-Sight opens the Edit menu.
5. From the Edit menu, select a command and click ✕.

COGNEX In-Sight

◄ ▲ ► Worksheet: Select a Row or Column

You select an entire row or column before performing a worksheet operation through the Edit menu. Control Pad conventions for multi-cell selection are:

Select entire row

Select
entire
column

Extend
range
right

Extend range down
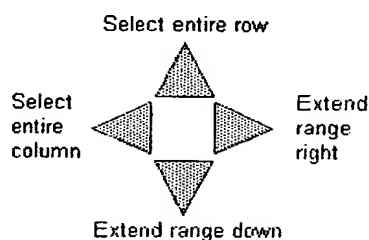
To select a range, see *Selecting a Cell or Cell Range*.

► **To Select an Entire Row or Column**

- Keyboard cursor = Control Pad cursor
- Shift-cursor = block selections
- In the Formula Builder, you can select functions by typing their names, which are case-insensitive. Formula Builder recognizes a function as soon as you type enough characters to uniquely identify it.
- F2 = □ =
- F3 = ○ = Edit menu (so does shift-up-arrow (then release)
- Enter = ✕ button
- Esc = △ button
- F2 + enter = trigger

COGNEX In-Sight

◄ ▲ ► Server: Commands

The following commands are available at the Server prompt.

Syntax	Description
acquire	Acquire image and run job.
clear *cellrange*	Clear cells.
getactive *cellrange*	Get enable state of cells.
getcell *cellrange*	Get cell definitions.
getdim	Get spreadsheet dimensions.
getmax	Get max row and col of cells.
getvalue *cellrange*	Get output value of cells.
help	Gets a list of commands.
setactive *cellnum string*	Set enable state of cells.
setcell *cellnum string*	Set cell definitions. You can enter ?strings?, numeric values, or valid formulas.
timeout *x*	Milliseconds until command timeout.
trigger	Get output value of cells.
update	Run job.

Parameters are:

- *cellnum*. Coordinates of cell in the spreadsheet. Case-insensitive. For example, c7.
- *cellrange*. Coordinates of a cell or range of cells. For example, c7 or c7:e13
- *string*. A character string that defines the contents of the cell.

COGNEX In-Sight

◄ ▲ ► Acquire: Functions Reference

The Acquisition functions and structures, automatically present on the worksheet in row 0, represent the image and the external input sources that can force the worksheet to update. For example, AcquireImage(), located in cell A0, acquires an image—a fundamental step for any application. These functions do not appear in the Formula Builder function list because they have dedicated cells in row 0.

This section discusses the following topics:

- Getting Started with Acquisition
- Selecting a Lens
- Field-of-View Table
- Lighting Guidelines
- Lighting Techniques
- Controlling Variable Lights
- Controlling Exposure
- Acquisition Reference.

In-Sight stores the acquired image an Image structure, used as the default image data source by many other vision processing functions.

COGNEX In-Sight

◄ ▲ ► Acquire: Getting Started

A typical inspection application acquires an image in response to an external trigger signal connected either to the camera's trigger inputs or to the Vision Processor's discrete or serial inputs. This topic describes how to configure In-Sight for acquisitions triggered through the camera. Because acquisition is an essential part of a vision application, In-Sight places the acquired image in the top row of the worksheet:

A0=	AcquireImage(B0,33,0,480,0,0,0,0,32,0)			
	A	B	C	D	E
0	Image				

Image structure in A0 stores the acquired image. AcquireImage() acquires the image and controls the camera and lights

You cannot move or delete cell A0, since most vision functions rely on the default image by default.

► **To Acquire an Image**

1. Choose and install a C-mount lens. For suggestions, see *Selecting a Lens*.

2. Choose and install a light source. For suggestions, see *Lighting Guidelines*.

3. Connect a trigger source to the camera's trigger inputs. Common hardware triggers include photo eyes, proximity sensors, and PLDs.

4. Place a representative object under the camera and obtain a preliminary image:
    a. Click and release the △ button, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the object location, camera height, aperture, and focus until satisfied with the image.
    b. Click △ twice to exit live mode, acquire the last image, and return to the worksheet. To manually acquire a new image at any time, hold □ and click ✕.

5. Move to cell A0, which contains an Image structure that stores the acquired image. Click ✕ to open the AcquireImage() property sheet, through which you define values that that control the camera and lights.

6. ??Set the trigger source. options include camera, continuous, manual, discrete, serial port 0, serial port 1, a timer, and software triggers defined in the worksheet. To specify the camera's trigger input as the trigger source, select **Source**, click ✕ to open the list, and then select **camera**. Click OK to close the property sheet.

7. Set the exposure time. For this exercise, increase the value from the default to 100. In an application, the shutter speed depends on part speed, inspection rate, light level, and lens aperture. Other image-related values set through AcquireImage() include the first scan line to acquire, the number of scan lines to acquire, the video gain (contrast), and video offset (brightness).

8. Click **OK** to close the AcquireImage() property sheet.

9. Apply a trigger signal through the camera's trigger inputs. In-Sight responds by acquiring an image and displaying it on the monitor.

═══ **COGNEX In-Sight**

◀ ▲ ▶ Acquire: Selecting a Lens

Selecting a lens is an important part of creating a successful machine vision application, but a detailed discussion of optics is beyond the scope of this manual. General factors influencing the choice include:

- The desired *field-of-view*, which is the maximum visible area seen through the lens. Basically, the field-of-view must be large enough to include all features of interest plus an adequate positional tolerance. It must be small enough to avoid wasting resolution on unimportant objects. See the *Field-of-View Table*.
- The *working distance* from the camera to the object. Physical or mechanical constraints often influence the minimum practical working distance. For the same field of view, increasing the working distance requires increasing the focal length of the lens, which reduces depth of field. Sometimes you can use mirrors or prisms to improve camera placement.
- The *physical size of the features*. Small features dictate a short working distance, a long focal length, or both.
- The *pixel resolution* required for inspection. The image processing and feature extraction algorithms

in the application depend on receiving an adequate amount of feature data from the image. The important features must span a sufficient number of pixels or else the software cannot reliably recognize them. Keeping the field of view as small as practical generally increases the size of the features in the image.

For information about Cognex C-mount lenses, see *Lens Reference*.

COGNEX In Sight

◄ ▲ ► Acquire: Field-of-View Table

Start with either the length of the field-of-view or the size of the smallest feature, both in inches. Then look up the focal length (16, 25, 50, or 75 mm), working distance (Dist, in inches), and extension-tube size (Ext, in millimeters) that supports the measurement. When possible, favor shorter working distances to increase resolution and shorter extension tubes to decrease lens aberrations. Note that this table ignores depth of field, which depends on focal length, working distance, and aperture.

FOV	Minimum Feature	16 mm	25 mm	50 mm	75 mm
0.30"	0.0018"	—	—	—	Dist: 11.5" Ext: 65.0 mm
0.35"	0.0021"	—	—	—	Dist: 11.5" Ext: 55.0 mm
0.40"	0.0024"	—	—	—	Dist: 12.0" Ext: 45.0 mm
0.50"	0.0030"	—	—	—	Dist: 12.5" Ext: 35.0 mm
0.60"	0.0036"	—	—	Dist: 9.0" Ext: 20.0 mm	Dist: 13.5" Ext: 26.0 mm
0.70"	0.0042"	—	—	Dist: 9.5" Ext: 16.0 mm	Dist: 14.5" Ext: 22.0 mm
0.75"	0.0045"	—	—	Dist: 10.0" Ext: 16.0 mm	Dist: 15.0" Ext: 20.0 mm
0.80"	0.0048"	—	—	Dist: 10.0" Ext: 15.0 mm	Dist: 15.5" Ext: 18.0 mm
0.90"	0.0054"	—	—	Dist: 11.0" Ext: 12.0 mm	Dist: 16.5" Ext: 15.0 mm
1.00"	0.0060"	Dist: 4.0" Ext: 3.0 mm	Dist: 5.5" Ext: 5.0 mm	Dist: 11.5" Ext: 11.0 mm	Dist: 17.5" Ext: 13.0 mm
1.25"	0.0075"	Dist: 4.5" Ext: 3.0 mm	Dist: 6.5" Ext: 4.0 mm	Dist: 13.5" Ext: 8.0 mm	Dist: 20.0" Ext: 10.0 mm
1.50"	0.0090"	Dist: 5.0" Ext: 2.0 mm	Dist: 7.5" Ext: 4.0 mm	Dist: 15.0" Ext: 7.0 mm	Dist: 22.5" Ext: 7.0 mm

1.75"	0.0099"	Dist: 5.5" Ext: 2.0 mm	Dist: 8.5" Ext: 3.0 mm	Dist: 17.0" Ext: 5.0 mm	Dist: 25.0" Ext: 5.0 mm
2.00"	0.0120"	Dist: 6.0" Ext: 2.0 mm	Dist: 9.5" Ext: 2.5 mm	Dist: 19.0" Ext: 5.0 mm	Dist: 28.0" Ext: 3.0 mm
2.25"	0.0132"	Dist: 7.0" Ext: 1.0 mm	Dist: 10.0" Ext: 2.0 mm	Dist: 21.0" Ext: 5.0 mm	—
2.50"	0.0147"	Dist: 7.5" Ext: 1.0 mm	Dist: 11.0" Ext: 1.5 mm	Dist: 23.0" Ext: 5.0 mm	—
2.75"	0.0162"	Dist: 8.0" Ext: 1.0 mm	Dist: 12.0" Ext: 1.5 mm	—	—
3.00"	0.0177"	Dist: 8.5" Ext: 1.0 mm	Dist: 13.0" Ext: 1.0 mm	—	—
3.50"	0.0204"	Dist: 9.5" Ext: 1.0 mm	Dist: 15.0" Ext: 1.0 mm	—	—
4.00"	0.0234"	Dist: 11.0" Ext: none	Dist: 16.5" Ext: 1.0 mm	—	—
4.500"	0.0264"	Dist: 12.0" Ext: none	Dist: 18.5" Ext: 1.0 mm	—	—
5.00"	0.0294"	Dist: 13.0" Ext: none	Dist: 20.0" Ext: 1.0 mm	—	—
6.00"	0.0351"	Dist: 15.5" Ext: none	Dist: 24.0" Ext: none	—	—
7.00"	0.0411"	Dist: 18.0" Ext: none	Dist: 28.0" Ext: none	—	—
7.50"	0.0439"	Dist: 19.0" Ext: none	Dist: 30.0" Ext: none	—	—
8.00"	0.0468"	Dist: 20.5" Ext: none	—	—	—
9.00"	0.0528"	Dist: 22.5" Ext: none	—	—	—
10.00"	0.0585"	Dist: 25.5" Ext: none	—	—	—

For information about Cognex C-mount lenses, see *Lens Reference*.

COGNEX In-Sight

◄ ▲ ► Acquire: Lighting Guidelines

Lighting is vitally important for machine vision applications. The general goal is to make the important features plainly visible in the image, so that the image processing and feature detection algorithms can start with the best possible pixel data. Specific goals might include freezing motion, increasing edge sharpness, minimizing or creating shadows, removing or creating reflections, and increasing the contrast between feature pixels and background pixels.

The optical properties of the feature and background surfaces strongly influence the lighting technique likely to yield good results. Smooth, reflective, specular surfaces differ from rough, dull, diffuse surfaces; flat surfaces differ from three-dimensional surfaces; etched differs from polished; copper differs from solder; and so on. One place to start when evaluating lights is to identify any differences in color, texture, or other optical properties that distinguish the features from the background. You can then select lights that exploit the differences to make the features more prominent. For example, text etched into a mirror would scatter light, but its background would reflect it.

The goal of the application also influences lighting. An application that measures the size of a three-dimensional part should avoid casting shadows that might cause confusion during measurement. On the other hand, an application that simply detects the presence of the part might benefit from conspicuous shadows that are absent when the part is absent. The solution depends on the purpose.

For information about Cognex lights, see *Lights Reference*.

COGNEX In-Sight

◄ ▲ ► Acquire: Lighting Techniques

Light can be white or colored, diffuse or directional, on-axis or off-axis, oblique or perpendicular, unidirectional or omnidirectional, and continuous or strobed. The wide range of choice makes it difficult to offer generalized suggestions. General techniques and rules of thumb include:

- **Ambient lighting** is usually undesirable, partly because it is uncontrolled and often strongly variable. Another problem is that both florescent and incandescent sources tend to oscillate at the 50 or 60 Hz line frequency. Consider isolating the subject from ambient light, so that the controlled source dominates the light used for imaging.
- **Back lighting** provides strong contrast between object and background, creating a silhouetted image appropriate for algorithms like blob analysis that require a near-binary image. On the other hand, it eliminates gray levels and surface detail.
- **Brightfield lighting** uses specular reflection to highlight smooth, reflective areas and darken rough textured areas. To create a brightfield effect, place the light on (or near) the lens axis:

Light   Lens

Rough surface
(dark feature)

Smooth surface
(bright background)
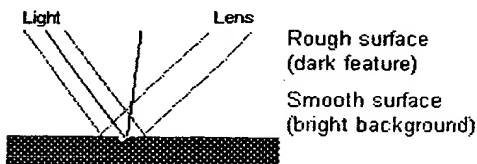
Smooth surfaces reflect light into the lens and appear bright; rough surfaces scatter light and appear dark. (The mirror image of the light itself is out of focus, effectively diffusing it.) A potential problem with brightfield lighting is excessive glare from the large amount of light reflected into the camera.

- **Darkfield lighting** uses specular reflection to darken smooth surfaces and brighten rough areas. To create a darkfield effect, place the light off the lens axis:

Light   Lens

Rough surface
(bright feature)

Smooth surface
(dark background)
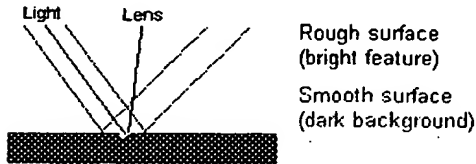
Most of the light reflects from the smooth surface away from the lens, but some of the light striking rough features reflects into the lens. Darkfield lighting highlights edges on a dark background. Only part of the light striking the feature scatters into the lens, so darkfield requires relatively high light levels.

- **Directional front lighting** creates distinctive shadows and reflections from three-dimensional, free-standing subjects. Different angles—from oblique to high-angle to coaxial—yield different effects.
- **Diffused front lighting** is good for inspecting high-contrast parts but poor for inspecting low-contrast, low-profile parts.
- **Omnidirectional lighting** (such as ring lighting) provides even illumination and eliminates shadows. It is often good for close inspections or for features with significant relief. It is poor for low-contrast subjects.
- **Oblique side lighting** can be used to cast shadows from features that have relief but lack contrast.
- **Colored or filtered light** can increase or reduce contrast. Complimentary color tends to increase contrast (good for pale features), and similar color tends to reduce contrast (good for dense features). For example, red light darkens blues and greens, so you might use a red light to separate a blue object from a red background.

For information about Cognex lights, see *Lights Reference*.

COGNEX In·Sight

◄ ▲ ► Acquire: Controlling Variable Lights

The In-Sight hardware lets you dynamically control up to four independent banks of lights. To do so set the AcquireImage function's LightPower values.

COGNEX In·Sight

◄ ▲ ► Acquire: Controlling Exposure

The amount of light that reaches the image sensor determines the brightness of the image. As you might

expect, increasing the exposure makes the image brighter. Finding an appropriate exposure depends on the details of your application, making it difficult to offer suggestions applicable in all situations. In general, you are seeking an exposure level that makes the important details easily visible in the image. One strategy is to place the features of interest near the center of the available grayscale range, seeking the highest-contrast image that does not saturate the brightest or darkest values. If you cannot digitize the entire brightness range without saturation, then skew the image toward the salient detail.

Several factors—light level, lens aperture, exposure time, and video gain—interact with each other to control exposure. Increasing or decreasing each parameter changes the image in a distinctive way. Whether the change is an advantage or a drawback depends on your application. Tradeoffs include:

- Increasing the light level increases the exposure but is sometimes impractical. Nonetheless, providing adequate light is almost always the first-choice solution for exposure problems.
- Increasing the aperture increases the exposure but decreases depth-of-field and lens sharpness. The shallow depth-of-field associated with long focal lengths or short working distances can be a problem for objects with significant depth or working distances with significant variability. In general, try to obtain generous depth-of-field unless you are intentionally blurring unwanted foreground or background detail. Try to use middle to upper-middle apertures when possible, avoiding the widest and smallest apertures.
- Increasing the exposure time increases the exposure but risks blur due to motion. Applications involving high part speeds or inspection rates require the use of a brief exposure or a strobe light to freeze motion.
- Increasing the video gain makes the image sensor more sensitive to light but can increase the visibility of video noise, making the image grainy. You set gain and a related parameter called offset through the AcquireImage function. The two parameters have the following relationship:

output = (input x gain) + offset

Increasing gain increases contrast (the range between the brightest and darkest pixels). Increasing offset increases brightness by shifting all grayscale values up by the same amount. Ignoring the fact that grayscale values saturate at the minimum and maximum values (0 and 255), offset does not change the range.

As a general rule of thumb, first add light, then add aperture, then add gain.

_____ **COGNEX In Sight**

◄ ▲ ► Acquire: Function Reference

AcquireImage	Acquires an image and stores it in an Image structure. Also controls the trigger source, variable lights, and the camera's analog-to-digital converter.
Image structure	Stores an image. Most vision processing functions by default refer to the acquired image, generated by AcquireImage() and stored in cell A0.

COGNEX In-Sight

◄ ▲ ► Acquire: AcquireImage function

Description	Acquires an image and stores it in an Image structure. Also controls the trigger source, variable lights, and the camera's analog-to-digital converter.
Heading	None. Always in cell A0.
Syntax	AcquireImage(Trigger, Manual, Exposure, Start Row, Num Rows, Light Power, Gain, Offset)

Inputs	**Trigger**	Trigger source. An image is acquired at the specified internal or external event. To define a soft trigger, see SetEvent.
	Manual	Enables the manual trigger (Control Pad□ +✕). When enabled, the manual trigger does not preempt the specified Trigger. AcquireImage() consequently acquires an image at either the manual trigger or the specified Trigger.
	Exposure	Exposure time, in milliseconds. Range, 0 to 1000. Default = 33, so that the brightness of live images matches that of acquired images. Internally rounded to the duration of one scan line (64 microseconds). For more information, see Controlling Exposure.
	Start Row	The first image row to acquire. Default = 0. ??Currently disabled??
	Num Rows	The number of image rows to acquire. Valid range, 1 to 480. Default = 480. ??Currently disabled??
	LightPower 0–3	Brightness value for light banks 0, 1, 2, or 3. Valid range, 0 through 255, where 0 is OFF, 255 is full brightness, and intermediate values are intermediate light levels. See Controlling Variable Lights. NOTE If you are using a Cognex strobe powered through the light control port, you must set the Light Power value to 255. ?? Confirm final behavior through RR??
	Gain	Raw video gain (contrast) for the image sensor. Gain interacts with Offset as follows: *Output = (input x gain) + offset* Valid range, 0 through 255. For more information, see Controlling Exposure.
	Offset	Raw video level (brightness) for the image sensor. Valid range, 0 through 255. For more information, see Controlling Exposure.
Stores	**Image**	On success, an Image structure, which stores an image.
Emits	Nothing.	
Errors	None.	

Comments	AcquireImage() acquires an image and creates an Image structure in cell A0 to store the image. You cannot move AcquireImage() from A0 or create a second AcquireImage() function. When AcquireImage() is enabled (the default) and a trigger occurs, AcquireImage() grabs a new image from the camera and updates A0.
	Accumulate, Count, or Latch cells sometimes use AcquireImage() as a signal to update their contents.
See Also	Image structure, Event, SetEvent.

COGNEX In-Sight

◄ ▲ ► Acquire: Image structure

Description	Stores an image. Most vision processing functions by default refer to the acquired image, generated by AcquireImage() and stored in cell A0.

Values	**Value**	**Description**	**Access function**
	Greyscale	Greyscale value, 0.00 through 255.00. Available for each pixel, specified by its row and column address.	GetPixelValue

Comments	Many vision processing functions rely on an Image structure as a data source. Nearly all by default refer to the acquired image, generated by AcquireImage and stored in cell A0.
	CompareImage, NeighborFilter, PixelFilter, and WarpImage modify pixel data and therefore store a new Image structure containing the modified image. To process an alternative image instead of the acquired image, change the Image reference from A0 to the new Image structure.
See Also	AcquireImage.

COGNEX In-Sight

◄ ▲ ► Blob: Overview

A *blob* is a contiguous patch of pixels with a distinctive grayscale value. Blob analysis, also called connectivity analysis or connected components analysis, is a feature extraction method for finding, analyzing, and classifying blobs in an image. It is often used to detect natural or organic shapes—ink dots, solder blobs, French fries, chicken wings, and so on—that are topologically stable but plastic, rotated, or both. Apples rolling on a conveyor belt, for example, exhibit various ovoid shapes within a known profile (area range in pixels, perimeter range in pixels, number of internal holes, and so on). An apple inspection application might use blob analysis to exclude objects that fail to fit the profile.

This section introduces In-Sight's Blob functions (located under Vision Processing), which offer your application information about the number, location, shape, and orientation of blobs. It discusses the following topics:

- Getting Started with Blob Analysis
- Using the Blob Functions
    - Extracting a Blob Database
    - Finding Blobs in the Database
    - Sorting the Database
    - Getting Blob Results
- Blob Reference.

In-Sight's blob functions offer automatic thresholding, the ability to select blobs by color, size, area, and other characteristics, and a choice of one-step or multi-step blob analysis. The one-step method, with one function that extracts blobs, sorts them by largest area, and returns a specified number of results, is straightforward but less flexible and less efficient. The multi-step method, with separate functions for the each stage, is more complex but flexible and efficient (it lets you re-sort and re-score the database without rebuilding it).

COGNEX In-Sight

◄ ▲ ► Blob: Getting Started

In-Sight offers both one-step and multi-step blob analysis. This topic demonstrates the basic operation of the one-step method. Completing this exercise gives you hands-on experience with In-Sight's blob analysis functions.

► **To Extract Blobs from an Image**

1. Acquire an appropriate image. For this exercise:
    a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
    b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*.

Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.

c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

To acquire a new image at any time, hold □ and click ✕.

2. Leaving cell **A1** empty, move to cell **A2** and click ✕ to open the <u>Formula Builder</u>.
   NOTE Try to leave a blank row above functions that emit formulas. In-Sight prints descriptive labels above emitted formulas if it has room for the labels. If the row above is occupied, it emits only the numeric values, without labels.

3. From the **Vision Processing** category, select **Blob**, and then open the <u>ExtractBlobs</u> property sheet.

4. To define the region of interest, select the **Region** heading and click ✕. In-Sight hides the worksheet and displays the Region cursor, which you interactively <u>adjust</u> with the Control Pad. Surround the area containing the part sample, which has four round holes and one larger rectangular hole. Click ✕ to return to the property sheet.

5. You define the expected characteristics of a blob by setting parameters on the ExtractBlobs() property sheet. Initially, try the following values:

   o **Number To Find,** the number of blobs to extract. Increase this value to "4", so that you can get results about more than one blob.
   o **Blob Color,** the color of the blob. Select **Either,** to extract both White and Black blobs.
   o **Background Color.** If the background of your image is mostly dark, select **Black;** if it is mostly light, select **White.**
   o **Area Limit: Max,** the largest number of pixels in the blob. Try setting this value at 5,000 pixels (probably small enough to exclude the rectangular blob).

For now, leave the other values unchanged.

6. Extract the blobs by clicking **Run.** ExtractBlobs() analyses the image for blobs that fit the selection criteria, then draws an outline around them:

Region of interest
Blob 0
Blob 2
Blob 3
Blob 1
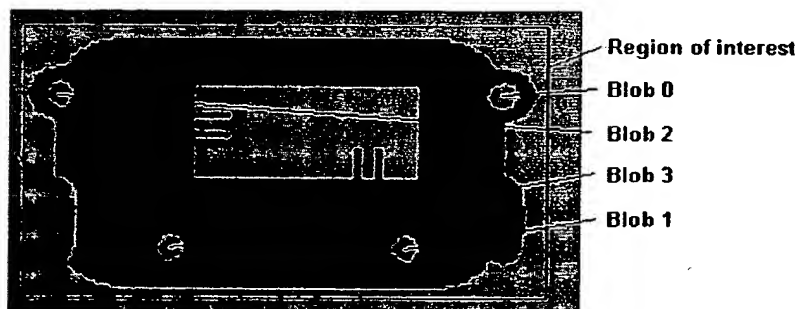
Even though the original image is grayscale, the blobs in the database are binary, because the image is segmented at a threshold value. Each Blob has an index, starting with zero, used to refer to it when the results contain more than one blob.

You can experiment with different selection criteria by changing their values. For example, you could increase the Area Limit value to include or exclude the larger rectangular blob. Each time you click

Appendix I - Page **124**

Run, ExtractBlobs() updates the blob database and blob outlines. Restrictive values tend to yield fewer blobs, and permissive values tend to yield more blobs.

7. **Examine the results.** When done experimenting, click **OK** to accept the current parameter values. For each blob, ExtractBlobs() emits the X and Y location of the centroid, the angle of the axis of the minimum moment of inertia, and other result values to the worksheet:

	Index	Row	Col	Angle	Color	Score	Area	Elongation	Hole
5 Blobs	0.000	244.406	352.656	89.856	1.000	0.000	14762.000	0.021	
	1.000	214.928	510.272	139.831	1.000	0.000	540.000	0.000	
	2.000	349.802	431.403	179.106	1.000	0.000	499.000	0.000	
	3.000	345.742	240.283	169.193	1.000	0.000	477.000	0.000	

To use a result value in a formula, create a reference to its cell. If you don't need one of the automatically generated result cells, you can delete it.

8. By default, In-Sight displays the blob graphics only when you highlight the Blobs structure, as described in *Flyover Graphics*. To display the blob graphics all the time, set the ExtractSortBlob() function's **Show** parameter to **Show All**.

ExtractBlobs() always sorts blobs by maximum area. To sort them by other criteria, you use the multi-step method.

_____ **COGNEX** In-Sight

◄ ▲ ► Blob: Multi-step Method

In-Sight offers both one-step and multi-step blob analysis. The one-step method combines several intermediate operations, simplifying the operation but reducing flexibility. The multi-step method separates the underlying operations, giving you more control but increasing complexity.

This section describes the typical sequence of operations:

1. Extracting a Blob Database (ExtractBlobs)
2. Finding Blobs in the Database (FindBlobs)
3. Sorting the Database (SortBlobs)
4. Getting Blob Results.

Separating the steps (building the database, scoring it, and sorting it) avoids the need to the database for each search, improving flexibility and efficiency.

_____ **COGNEX** In-Sight

◄ ▲ ► Blob: Extracting Database

The first step for blob analysis is to define the features that distinguish a blob from its background. You define blob characteristics through ExtractBlobs, which builds a blob database when you click **OK** or **Run** on the property sheet. The resulting Blobs structure stores the database, for subsequent processing by FindBlobs, SortBlobs, or TransBlobToWorld.

Extraction criteria include:

- **Threshold**: a grayscale value, from 0 to 63, used to recognize background pixels. If the background is "White," all pixels with grayscale values below the threshold are background pixels. If the background is "black," pixels above the threshold are background pixels. You can enter a threshold value manually or find the best value automatically by specifying −1, which automatically computes the threshold that generates the best contrast between object and background.

- **Background Color**: Sets the background color, either black or white.

- **Blob Color**: Sets the blob color: black, white, or either,

- **Include Holes**: *Holes* are background-colored pixels inside the blob. Choose OFF to exclude holes from blob statistics, or ON to include them.

- ??Expand this table to include the others.

After defining the profile you expect for a blob, click **OK** or **Run** on the property sheet. ExtractBlobs() then performs connectivity analysis using the specified selection criteria. It stores the resulting database in a Blobs structure, emits the number of blobs in the database, and draws a result graphic highlighting the blobs and their indices:
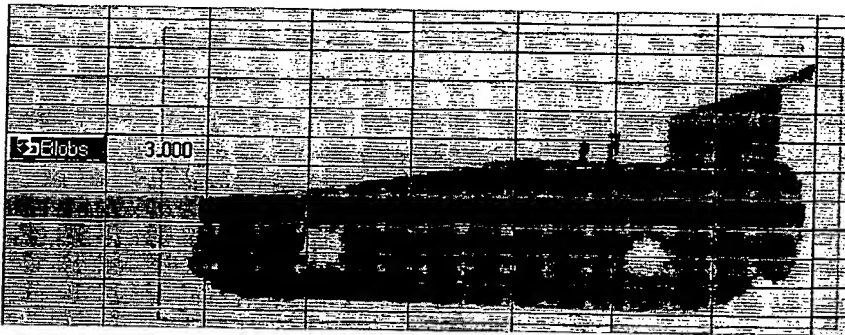
Even if the original image is grayscale, the database itself is binary.

After a creating a database, you can find blobs in it, sort it, or both.

COGNEX In-Sight

◄ ▲ ► Blob: Finding Blobs

After building a Blob database, you can find blobs in it with the FindBlobs function. Basically, this operation computes a score for every blob in the database. The score value depends on several criterion, each with an assignable weight, allowing you to exactly define the relative importance of each criterion.

Criterion	Description
Number to Find	The number of blobs to find in the database. Find stops scoring when it reaches this value.
Acceptance Threshold	A minimum acceptable score, 0 through 100.
Angle	xxxx
Area	The area, in pixels, of the blob.
Elongation	The elongation value—the ratio of the moments about the principal axes of inertia.
Holes	The number of holes within the blob.
Perimeter	The number of pixels on the perimeter of the blob.
Spread	The spread value—a rough measure of compactness.

Specifying "narrow" or "wide" ranges naturally tends to decrease or increase the number of results returned from the database.

COGNEX In-Sight

◄ ▲ ► Blob: Sorting Database

Sorting the database lets you rearrange it, selectively eliminating blobs. Sorting the database is an optional step, but you can sort more than once, to progressively prune the database.

Parameters are:

- Number to Sort. ??Throws out the rest.
- Sort By. The criterion for sorting. ??Give the list of values??
- Reference: Sets the location of the reference point, used when Extract is "Distance." You can set the reference point either by entering an absolute pixel grid coordinate into the Row and Col text boxes, or by clicking Cursor and dragging the cross-shaped cursor in the image.

COGNEX In-Sight

◄ ▲ ► Blob: Getting Result Values

ExtractBlobs creates Blobs structure that stores the blob database. FindBlobs, SortBlobs, and TransBlobToWorld refer to this Blobs structure as a data source and generate a second Blobs structure to hold the modified database. They also emit useful data from the Blobs structure to the worksheet, for use by formulas that process the blob results:

	Index	Row	Col	Angle	Color	Score	Area	Elongation	Hole
Blobs	0.000	244.406	352.655	89.855	1.000	0.000	14762.000	0.021	
	1.000	214.928	610.272	139.831	1.000	0.000	540.000	0.000	
	2.000	349.802	431.403	179.106	1.000	0.000	499.000	0.000	
	3.000	345.742	240.283	169.193	1.000	0.000	477.000	0.000	

For details about the emitted values, see *Blobs Structure*.

Any formula can refer to an emitted formula to obtain its value for further processing. If you don't need a result, you can throw away the emitted formula.

COGNEX In·Sight

◄ ▲ ► Blob: Function Reference

Blobs structure	Stores a database of blobs, each with a zero-based index and a profile of measured values. Coordinates are in Image units (pixels) unless converted by TransBlobToWorld.
ExtractBlobs	Builds a blob database, optionally sorting it. Stores its results in a Blobs structure. Emits a specified number of blobs, sorted by largest area.
FindBlobs	Scores blobs in a Blobs database by closeness to ideal values, each with an assignable weight. Stores a new, scored Blobs structure.
SortBlobs	Sorts a Blobs structure by specified criteria. Stores a new, sorted Blobs structure.

TransBlobToWorld, under Coordinate Transforms, converts blob results to World coordinates. Results functions, under Data Access, that get results from a Blobs structure are GetAngle, GetArea, GetCol, GetColor, GetElongation, GetHoles, GetMaxCol, GetMaxRow, GetMinRow, GetNFound, GetPerimeter, GetRow, GetScore, GetSpread.

COGNEX In·Sight

◄ ▲ ► Blob: Blobs structure

Description	Stores a database of blobs, each with a zero-based index and a profile of measured values. Coordinates are in Image units (pixels) unless converted by TransBlobToWorld.		
Values	**Value**	**Description**	**Access function**
	Index	Zero-based index identifiying a particular blob. The first blob is indexed 0.00, the second is 1.00, and so on.	—
	Angle	Angle of the major axis of elongation, in CCW degrees. Measured from the image row axis unless converted to World coordinates by TransBlobToWorld(). NOTE The angle value ranges from 0.00 through 180.00. It does not distinguish the "direction" of the blob. For example, a blob with a 90-degree rotation reports a 90-degree angle. The same blob rotated 180 degrees also reports a 90-degree angle, not a 270-degree angle. If you must distinguish the absolute direction, consider using pattern matching instead of blob analysis.	GetAngle
	Area	Number of pixels inside the blob. ??Always in pixels even if the Blobs structure is converted to World units. ??Includes holes if XXX; excludes holes if YYY??	GetArea
	Column	Centroid column coordinate, ??offset by reference point??. In pixels unless converted to World units by TransBlobToWorld().	GetCol
	Color	Blob color, relative to the threshold. 0.00 for a black blob, and 1.00 for a white blob.	GetColor
	Elongation	Ratio of the major axis to the minor axis.	GetElongation
	Holes	Number of background-colored areas inside the blob.	GetHoles
	MaxCol	Maximum column coordinate. The blob's right-most point. In pixels even if the Blobs structure is converted to World coordinates (because the value in World coordinates might be a completely different point on the blob perimeter). Not emitted.	GetMaxCol
	MaxRow	Maximum row coordinate. The blob's top-most point. In pixels even if the Blobs structure is converted to World coordinates (because the value in World coordinates might be a completely different point on the blob perimeter). Not emitted.	GetMaxRow
	MinCol	Minimum column coordinate. The blob's left-most point. In pixels even if the Blobs structure is converted to World coordinates (because the value in World coordinates might be a completely	GetMinCol

Appendix I - Page **129**

		different point on the blob perimeter). Not emitted.	
MinRow		Minimum row coordinate. The blob's "bottom" point. In pixels even if the Blobs structure is converted to World coordinates (because the value in World coordinates might be a completely different point on the blob perimeter). Not emitted.	GetMinRow
NFound		Number of blobs in the database.	GetNFound
Perimeter		Number of pixels on the perimeter, summed along the blob boundary. Not emitted. ??Always in pixels even if the Blobs structure is converted to World units. ?A 1-pixel blob has a 4-pixel perimeter.?	GetPerimeter
Row		Centroid row coordinate. ?offset from the reference point?. In pixels unless converted to World units by TransBlobToWorld().	GetRow
Score		As computed by FindBlobs, 0.00 to 100.00.	GetScore
Spread		A rough measure of blob compactness.	GetSpread

Comments   None.
See Also   FindBlobs, SortBlobs, TransBlobToWorld

COGNEX In-Sight

◄ ▲ ► Blob: ExtractBlobs function

Description	Builds a blob database, optionally sorting it. Stores its results in a Blobs structure. Emits a specified number of blobs, sorted by largest area.
Heading	Vision Processing/Blob
Syntax	ExtractBlobs(Image, Fixture, Region, ..., Show)
Inputs	**Image** — Image data source, a reference to an Image structure. Default is A0.

	Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	Row and Column are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Region	Region of interest. **X** and **Y** define the top-left point by offsets from the
	—X	Fixture origin, in pixels. **High** and **Wide** are the height and width, in pixels.
	—Y	**Angle** is a rotation from the Fixture X axis, ± 360° CCW. See *Defining a*
	—High	*Region of Interest*.
	—Wide	
	—Angle	
	Num to Sort	Upper limit on the number of blobs returned. Zero (0) for unsorted.
	Threshold	Binary threshold, 0 through 255, to segment the image. Grayscale values the threshold are foreground pixels, and those > the threshold are background pixels. −1 to automatically compute the threshold that best separates the image into foreground and background regions (the optimum bimodal distribution).

Include Holes	ON includes background-colored holes in the blob statistics; OFF excludes them.
Boundary Blobs	ON includes blobs crossing or touching the Region edge; OFF excludes them. ??boundary blobs in interface??
Color	Blob color, relative to the threshold. One of **black**, **white**, or **either**.
—Blob —BG	Background color, relative to the threshold. One of **black** or **white**.
Area Limit —Min —Max	Minimum and maximum area, in pixels.
Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	On success, a <u>Blobs structure</u>, which stores a blob database.
Emits	• If **Num to Sort** = 0: the number found.
	• If **Num to Sort** > 0: Index, Centroid Row, Centroid Column, Angle, Blob Color, Score, Area, Elongation, Holes, Perimeter, and Spread.

For details, see <u>Blobs Structure</u>.

Errors	Invalid parameter.
Comments	None.
See Also	<u>FindBlobs</u>, <u>SortBlobs</u>, <u>TransBlobToWorld</u>, *Getting Started with Blob Analysis*.

COGNEX In·Sight

◄ ▲ ► Blob: FindBlobs function

Description	Scores blobs in a Blobs structure by closeness to ideal values, each with an assignable weight. Stores a new, scored Blobs structure.
Heading	Vision Processing/Blob
Syntax	FindBlobs(Blobs, Num to Find, *match-criteria*, ..., Show)
Inputs	**Blobs** A Blobs structure, created by <u>ExtractBlobs</u>.
	Num to Find Maximum number of matches.
	Accept Thresh Acceptance threshold—the minimum acceptable score, 0 through 100.
	Angle —Ideal —Range —Weight Angle of the major axis, in counterclockwise degrees. Range is a bracket, in ± degrees. Weight is importance in scoring, 0.00 to 100.00.
	Area —Ideal Area, in pixels. Range is a ± bracket, in pixels. Weight is importance in scoring, 0.00 through 100.00

Appendix I - Page 131

—Range	~~Ideal~~ ~~scor~~ ~~g, 0.00 through 100.00.~~
—Weight	
Elongation	Ratio of major axis to minor axis. Range is a ± bracket, centered on the
—Ideal	ideal ratio. Weight is importance in scoring, 0.00 to 100.00.
—Range	
—Weight	
Holes	Number of holes, or background-colored areas inside the blob. Range is a
—Ideal	± bracket. Weight is importance in scoring, 0.00 to 100.00.
—Range	
—Weight	
Perimeter	Perimeter, in pixels. Range is a ± bracket, in pixels. Weight is importance in
—Ideal	scoring, 0.00 to 100.00.
—Range	
—Weight	
Spread	A rough measure of blob compactness. Range is a ± bracket. Weight is
—Ideal	importance in scoring, 0.00 to 100.00.
—Range	
—Weight	
Spread	Ideal: A rough measure of blob compactness.
	Range is a ± bracket, in pixels
	Weight: Importance of Spread.
Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	On success, a <u>Blobs structure</u>, which stores a blob database.
Emits	For each blob: Index, Centroid Row, Centroid Column, Angle, Blob Color, Score, Area, Elongation, Holes, Perimeter, and Spread. For details, see <u>Blobs Structure</u>.
Errors	Invalid parameter.
Comments	None.
See Also	*Finding Blobs in the Database*, ExtractBlobs, SortBlobs, TransBlobToWorld

COGNEX In-Sight

◄ ▲ ► Blob: SortBlobs function

Description	Sorts a Blobs structure by specified criteria. Stores a new, sorted Blobs structure.
Heading	Vision Processing/Blob
Syntax	SortBlobs(Blobs, Num to Sort, Sort By, Reference, Show)

Inputs	**Blobs**	A reference to a Blobs structure, from <u>ExtractBlobs</u>, <u>FindBlobs</u>, or <u>SortBlobs</u>.
	Num to Sort	Number of blobs to sort.
	Sort By	The criterion for sorting, one of:

angle	Minimum to maximum.
area	Maximum to minimum.
distance	From the reference point.
elongation	The ratio of major to minor axes, ?smallest to largest?
grid	Rows, then columns, from the reference point.
holes	Maximum to minimum.
col	?Left to right?
Row	?Top to bottom?
perimeter	Maximum to minimum.
spread	A measure of compactness, ?smallest to largest?

Reference **—Row** **—Column** **—Angle**	Sort reference row and column, in pixels from the image origin. **Angle** is a rotation, ± 360 degrees CCW from the image row axis.
Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	On success, a <u>Blobs structure</u>, which stores a blob database.
Emits	For each blob: Index, Centroid Row, Centroid Column, Angle, Blob Color, Score, Area, Elongation, Holes, Perimeter, and Spread. For details, see <u>Blobs structure</u>.
Errors	Invalid parameter.
Comments	None.
See Also	*Sorting the Blob Database*, <u>ExtractBlobs</u>, <u>FindBlobs</u>, <u>TransBlobToWorld</u>

COGNEX In·Sight

◄ ▲ ► Clocked Data: Overview

The Clocked Data functions perform time-related processing. Several functions—Accumulate, Count, and Latch—keep various kinds of running totals. Another function—DelayLine()—stores the history of a value in a buffer, typically to delay an output signal to synchronize it with a device deeper in the production line.

This section introduces the Clocked Data functions, discussing the following topics:

- Getting Started with Clocked Data
- Clocked Data Reference.

NOTE To access the Clocked Data Storage functions from the Formula Builder, click **Advanced**.

COGNEX In-Sight

◄ ▲ ► Clocked Data: Getting Started

In this section, you create a DelayLine buffer that monitors the row result from a blob extracted from the image. The result looks like this:

A2=	DelayLine(A0,C1,10,0)					
	A	B	C	D	E	F
0	ⓑImage					
1	ⓑBlobs	0.000	357.666	419.988	167.265	1.000
2	ⓑDelay	357.684	357.665	0.018	357.636	357.684

Row centroid, current image

Delay object, stores the buffer

Mean of buffered values

Largest value in buffer

Buffered value, previous image

Standard deviation

Smallest value in buffer

▶ **To Create a Buffer**

1. Acquire an appropriate image. For this exercise:
    a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
    b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
    c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

    To acquire a new image, hold ☐ and click ✕.

2. Create the data to buffer. A convenient value for this exercise is the centroid row value from a blob:
    a. From cell **A1**, click ✕ to open the Formula Builder.
    b. From the Blob heading, open the <u>ExtractBlobs</u> property sheet.
    c. Leaving the defaults unchanged, click **OK** to close the property sheet. ExtractBlobs() builds and sorts a blob database in one step. It also emits various results, including the centroid row and column:

Blobs object, from ExtractSortBlobs()

Row centoid, emitted by GetRow()

C1=	GetRow($A$1,B1)			
	A	B	C	D
0	Image			
1	Blobs	0.000	357.659	419.994
2				

3. Move to cell A2 and click ✕
4. Click **Advanced** to view the **Clocked Data Storage** category.
5. Open the DelayLine() property sheet. Set the following values:
    o Leave the **Event, Num Steps,** and **Reset** values unchanged. By default, DelayLine() updates its internal buffer each time the system acquires an image.
    o Click the **Data** value, temporarily closing the property sheet so that you can interactively create a reference to a data source. For this example, select cell **C1**, the row centroid.
    o Click **OK** to close the property sheet. DelayLine() creates an empty buffer, stores it in a <u>Delay structure</u>, and emits a block of formulas to display values from the Delay structure. At this point, the buffer is empty, so the emitted values are all #ERR.
    o Click and release △ to open the System menu. Select **Triggers** to open the Triggers menu, and then click **Manual**. In-Sight acquires a new image and updates the buffer, which now contains one value.
    o Click **Manual** at least ten more times to fill the buffer with values:

A2=	DelayLine(A0,C1,10,0)				
	A	B	C	D	E	F
0	Image				
1	Blobs	0.000	357.666	419.988	167.265	1.000
2	Delay	357.684	357.665	0.018	357.636	357.684

Row centroid, current image

Delay object, stores the buffer

Buffered value, previous image

Mean of buffered values

Standard deviation

Largest value in buffer

Smallest value in buffer

Small differences in lighting usually cause the histogram values to change slightly from image to image. The buffer reflects these differences. After the buffer fills, the oldest value drops out. (You can set the number of time increments to buffer in the DelayLine() property sheet. The default, unchanged here, is 10.)
    o From the Triggers menu, enable **Continuous**. In-Sight acquires images as quickly as possible, updating the buffer to reflect them.

Appendix I - Page **135**

COGNEX In-Sight

◄ ▲ ► Clocked Data: Function Reference

Accumulate	Keeps a running total of a specified value, updating at a specified event.
Count	Starting at a specified integer, adds one to a running total at each event.
Delay structure	Stores a DelayLine() buffer.
DelayLine	Buffers the history of a value. Emits the current, last, mean, and standard deviation values.
Latch	Returns a specified value at a specified event.

Functions that return statistics from a Delay structure are GetCount, GetMax, GetMean, GetMin, GetSDev, GetSum, and GetValue.

COGNEX In-Sight

◄ ▲ ► Clocked Data: Accumulate function

Description	Keeps a running total of a specified value, updating at a specified event.	
Heading	Advanced/Clocked Data Storage	
Syntax	Accumulate(Event, Value, Reset, Preset)	
Inputs	**Event**	An event that forces an update of the accumulated value. Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	Value	The value to accumulate at each **Event**. Can be a cell reference. Positive to increment, negative to decrement.
	Reset	When ON, resets the current total to the **Preset** value, at the next **Event**.
	Preset	First value at startup or reset. An integer.
Returns	The current running total.	
Emits	Nothing.	
Errors	Invalid parameter.	
Comments	At creation or reset, the accumulator contains the **Preset** value.	
See Also	Count, DelayLine, Latch.	

COGNEX In-Sight

◄ ▲ ► Clocked Data: Count function

Description	Starting at a specified integer, adds one to a running total at each event.
Heading	Advanced/Clocked Data Storage
Syntax	Count(Event, Max Value, Reset, Preset)

Inputs	**Event**	An event that forces an update of the running total. Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	Max Value	The maximum value for the counter, an integer. When the counter reaches **Max Value**, the running count resets to **Preset** at the next Event.
	Reset	When on, resets the current total to the **Preset** value, at the next **Event**
	Preset	First value at startup or reset. An integer.

Returns	The counter value at the most recent **Event**.
Emits	Nothing.
Errors	Invalid parameter.
Comments	At creation or reset, the buffer contains one value—the current **data** value.
See Also	Accumulate, DelayLine, Latch.

COGNEX In-Sight

◄ ▲ ► Clocked Data: Delay structure

Description	Stores a DelayLine buffer.
Heading	Advanced/Clocked Data Storage

Values	**Value**	**Description**	**Access function**
	Count	Number of values in the buffer. When you define a buffer, you specify the number of time-intervals it will contain. At creation, the buffer contains only one value. As the buffer fills, the number of values increases until it reaches the specified depth. **Count** is the number of values actually in the buffer, even if the buffer is not full.	GetCount
	Maximum	The largest value currently in the buffer.	GetMax
	Mean	The arithmetical mean of the values in the buffer.	GetMean
	Minimum	The smallest value currently in the buffer.	GetMin
	Standard Deviation	The standard deviation of the values in the buffer.	GetSDev
	Sum	Sum of the values in the buffer.	GetSum
	Value	The value in the buffer at the specified time offset	GetValue
Comments	None.		
See Also	DelayLine		

COGNEX In-Sight

◄ ▲ ► Clocked Data: DelayLine function

Description		Buffers the history of a value. Emits the current, last, mean, and standard deviation values.
Heading		Advanced/Clocked Data Storage
Syntax		DelayLine(Event, Value, Num Steps, Reset)
Inputs	**Event**	An event that forces an update of the running total. Must be a reference to a cell containing <u>AcquireImage</u> (the default), <u>Button,</u> or <u>Event</u>.
	Value	The value to buffer—a cell reference or formula.
	Num Steps	Number of time increments to buffer.
	Reset	Resets the buffer, eliminating previously buffered values. After reset, the buffer contains one value, the current **Value**.
Stores	**Delay**	On success, a <u>Delay structure</u>, which stores the buffer.
Emits		Nothing.
Errors		Invalid parameter.
Emits		Current, mean, standard deviation, maximum, and minimum values. For details, see <u>Delay structure</u>.
Comments		At creation or reset, the buffer contains one value—the current **Value**.
See Also		<u>Accumulate</u>, <u>Count</u>, <u>Latch</u>, <u>*Getting Started with Clocked Data*</u>.

COGNEX In·Sight

◄ ▲ ► Clocked Data: Latch function

Description		Returns a specified value at a specified Event.
Heading		Advanced/Clocked Data Storage
Syntax		Latch(Event, Value)
Inputs	**Event**	An event that forces an update of the latched value. Must be a reference to a cell containing <u>AcquireImage</u> (the default), <u>Button,</u> or <u>Event</u>.
	Value	A numeric value for output.
Returns		The latch value at the most recent **Event**.
Emits		Nothing.
Errors		Invalid parameter.
Comments		At creation or reset, the buffer contains the current value.
See Also		<u>Accumulate</u>, <u>Count</u>, <u>DelayLine</u>.

COGNEX In·Sight

◄ ▲ ► Coordinate Transforms: Overview

The Coordinate Transforms functions map values from one coordinate system to another. They are grouped in three subheadings, corresponding to the major In-Sight coordinate systems:

Calibrate   Establishes a coordinate transformation between two planes. Often used to create a calibrated transformation, which converts in either direction between image locations (in Pixel coordinates) and physical locations (in World coordinates). For example, an application might convert points and distances from row-and-column pixel values to real-world units such as millimeters or inches. To simplify the process of converting results from feature extraction and measurement operations, In-Sight offers dedicated functions that convert a Blobs, Dist, Edges, or Patterns structure in one step.

Although through-the-lens calibration is a common use for coordinate transformation, you can use In-Sight's calibration functions for other plane-to-plane transforms. Because In-Sight computes the transform from four known points, the two coordinate systems can be skewed.

Fixture   Converts between a reference (Fixture) coordinate system and either World or Pixel values. Establishing a fixture coordinate system, based on a "landmark" feature identified in each image, compensates for variation in position or orientation between successive images. The Fixtured transforms map a fixtured point to the equivalent point in the current image.

Warp   Maps a point in a warped image (constructed by WarpImage) back to the original, unwarped image.

This section introduces the Coordinate Transforms functions, discussing the following topics:

- Getting Started with Calibration
- Converting Pixel Values to World Units
- Understanding Calibration
- Getting Started with Fixture Transforms
- Getting Started with Warp Transforms
- Coordinate Transforms Reference.

COGNEX In-Sight

◄ ▲ ► Calibration: Getting Started

To construct a *calibrated transformation*, you must know the locations of several points in the real world and the corresponding points in the image. The transformation obtained from the set of known points can then convert any point in one coordinate system to the other coordinate system through interpolation. A common technique for specifying the real-world locations is to construct a calibration plate, typically a grid with a known spacing. After acquiring an image of the calibration plate with the intended optical and physical setup, you can identify the corresponding points in both coordinate systems.

In this topic, you acquire an image of a sample calibration plate and use it to define a calibrated transformation. In the next topic, you use this calibration to convert values from image units (pixels) to world units (millimeters).

▶ **To Define a Calibration**

1. Display the sample calibration plate (dots on a 10-millimeter grid), print it, and return to this topic.
2. Place the printed calibration plate under the camera.
3. Click and release △ and select **Live** from the System menu, placing In-Sight in live mode. Adjust the focus and aperture, make the grid square in the image, and click △ In-Sight exits live mode, acquires the final image, and returns to the worksheet. The image should have good contrast without shadows, reflections, or other image defects.
4. Move to cell **A1** and click ✕ to open the Formula Builder.
5. From **Coordinate Transforms**, open the **Calibrate** heading, and then open the Calibrate property sheet.
6. On the property sheet, select **Pixel Point 0** and click ✕. By default, In-Sight assumes you want to set the image point locations interactively. Its draws the four calibration points on the image, highlighting the active point (point 0):
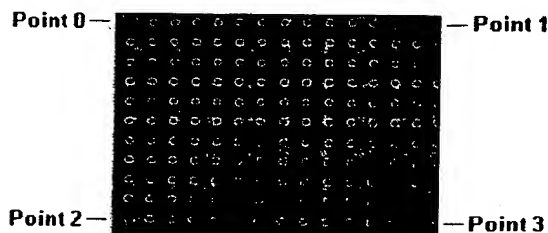
7. Using the Control Pad cursor, move the point graphic to the top-left dot in the calibration plate image. The graphic indicates the row and column position as follows:
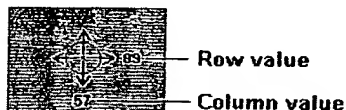
When done, click ✕ to accept the location.
8. On the property sheet, click **World Point O**. In-Sight displays its **X** and **Y** values. Edit **X** to 0.00 and **Y** to 0.00, defining it as the origin of the World coordinate system.
9. Click **Pixel Point 1**, drag the point graphic to tenth dot to the right of Point 0, in the same row. Click ✕ to accept.
10. Click **World Point 1**, edit **X** to 0.00 (because the points are in the same row, and the grid is square in the image), and edit **Y** to 100.00 (because each dot is 10 millimeters apart, so ten dots total 100 millimeters).
11. Click **Pixel Point 2**, drag the point graphic to the seventh dot below Point 0, in the same column, and click ✕ to accept.
12. Click **World Point 2**, edit **X** to 70.00 and **Y** to 0.00, and click✕ to accept.
13. Click **Pixel Point 3**, drag it to the dot seven down and ten right from Point 0, and click ✕ to accept.

14. Click **World Point 3**, edit **X** to 70.00 and **Y** to 100.00, and click ✕ to accept. The result should look about like this:
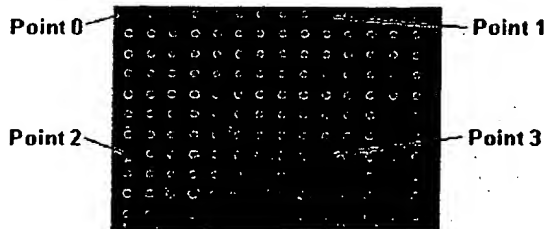
15. When done, click **OK** to accept the point values. Calibrate() returns a <u>Calib structure</u> that stores the calibration. Other functions that require calibration data subsequently refer to the Calib structure to obtain the calibration.
16. Go on to the <u>next topic</u>, where you apply this calibration to convert the values from image units to world units.

───────────────────────────────────── **COGNEX** In-Sight

◄ ▲ ► Calibration: Converting Values

In the <u>previous topic</u>, you defined a calibrated transformation. Here, you use it to convert the coordinates values in a Blobs structure from Image units (pixels) to World coordinates (millimeters). At the end, you convert a point from World coordinates back to Pixel coordinates.

▶ **To Convert Values with a Calibrated Transformation**

1. If you have not yet completed the <u>previous topic</u>, do so now.
2. Remove the calibration plate image and replace it with the <u>part sample image</u>. Hold ☐ and click ✕ to acquire a new image.
3. Get some feature data:
   a. Leaving cell **A2** empty, move to cell **A3** and click ✕ to open the Formula Builder.
      NOTE Try to leave a blank row above functions that emit formulas. In-Sight prints descriptive labels above emitted formulas if it has room for the labels. If the row above is occupied, it emits only the numeric values, without labels.
   b. From **Vision Processing**, open the **Blob** heading, and then open the <u>ExtractBlobs</u> property sheet.
   c. Select **Region**, click ✕, and interactively adjust the region of interest to surround the image of the part.
   d. Increase the **Number to Find** value to **4** and click **OK** to accept. ExtractBlobs() extracts four blobs, stores them in a Blobs structure, and emits various formulas:

Calib object (stores a calibrated transform)

Blobs object (stores a blobs database)

Region of interest

Emitted formulas

4. Convert the feature data to World units:

    a. Skipping cell **A7**, move to cell **A8** and click ✕ to open the Formula Builder. From **Coordinate Transforms**, open **Calibrate**, select <u>TransBlobToWorld</u>, and click ✕ to open its property sheet.

    b. Click ✕ on the **Calib** parameter, move the <u>marquee</u> to **A1** (the Calib structure), and click ✕ to accept the reference.

    c. Click ✕ on the **Blobs**, move the marquee to **A3** (the Blobs structure), and click ✕ to accept the reference.

    d. Leave **Num to Convert** unchanged at **3**.

    e. Click **OK** to close the property sheet. TransBlobToWorld() applies the calibrated transformation from **A1** to applicable values in **A3**. It stores a new Blobs structure in **A8** to hold the converted values. It emits the changed values—in this case, the centroid Row and Column values:

Image coordinates

World coordinates

World-to-pixel conversion

Here, the first blob's centroid (indexed 0.000) was offset 280.328 pixels from the image row axis. In the calibrated World coordinate system, the same point is offset 60.880 millimeters from the World origin.

Note that even though the original Blobs structure held four blobs, only three were converted. Also note that values such as Color and Score were not originally measured in image units. TransBlobToWorld() therefore did not convert them or generate the formulas that emit them.

Appendix I - Page **142**

5. Finally, convert one point back to its original pixel values:
   a. Move to **A11** and click ✕ to open the Formula Builder.
   b. From **Coordinate Transforms**, open **Calibrate**, select <u>TransWorldToPixel</u>, and click ✕ to open its property sheet.
   c. Click ✕ on **Calib**, move the marquee to **A1** (the Calib structure), and click click ✕ to accept the reference.
   d. Click ✕ on **Point** and move the marquee to **C8** (the row value for blob 0, expressed in World coordinates).
   e. Click and hold ◯, click cursor Right to select the range **C8:C9** (both the row and column values).
   f. Release ◯, and then click ✕ to accept the reference. Click **OK** to close the property sheet. TransWorldToPixel() converts the row and column values from World coordinates back to the original pixel values. **B11** should equal **C3**, and **B12** should equal **D3**, showing that a calibrated transformation is reversible.

COGNEX In-Sight

◄ ▲ ► Calibration: Understanding

A calibrated transformation maps locations in physical space, as defined by a <u>calibration plate</u>, to locations in an image:

Physical calibration plate (10 mm dot pitch) → **Acquired image of calibration plate** / **Image coordinate system** → **Image point (7.0 pixels, 8.0 pixels)** → **Transformation** → **Physical point (291 mm, 333 mm)**
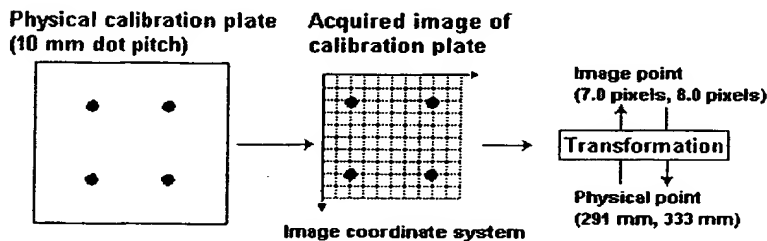
Calibration makes it easier for you to define a coordinate system that is linked to the physical world. You typically establish a calibration when setting up an application, updating it if you change some aspect of the physical or optical setup.

To perform a calibration, you acquire an image containing features that have known locations. You supply these locations to the <u>Calibrate</u> function *in both coordinate systems*. Calibrate() then constructs a transformation that maps points between image (Pixel) coordinates and physical (World) coordinates in either direction. When choosing the calibration points, observe the following rules of thumb:

- You can supply any convenient set of four points, provided you know their real-world locations. A grid-like arrangement is common but not required. The points cannot be collinear.
- The calibrated setup must be identical to the production setup. Differences in optics, <u>working distance</u>, and other factors invalidate the calibration.

- As much as practical, keep the lens and the object in the same plane. In-Sight's four-point calibration tolerates skew, but eliminating perspective distortion improves accuracy.
- As much as practical, keep the calibration plate square in the image.
- Limit the calibration to the image region that contains the features of interest. For example, if the important features are in the upper-left quadrant, then place the calibration points at its perimeter.
- Avoid extrapolating beyond the calibrated area.
- The underlying mathematics assume an ideal lens, without pincushion or barrel distortions. Lenses with significant geometric distortions reduce the accuracy of the calibration.

After you create a calibration, you can use it to transform data emitted by vision functions into precise physical locations.

COGNEX In-Sight

◄ ▲ ► Fixtured Transforms: Getting Started

A Fixture coordinate system is a reference coordinate system (X, Y, and theta) identified in an image and used to define other points or regions through offsets, typically to compensate for image-to-image variation in position, orientation, or both. With In-Sight, you can map a point from Fixture to Pixel coordinates in either direction with the TransPixelToFixture and TransFixtureToPixel functions. This topic describes how to use them.

▶ **To Create a Fixtured Transformation**

1. Acquire an image. During development, you typically place a part sample under the test stand and snap a picture of it. Alternatively, to adjust the camera height, aperture, and focus, use live mode.
2. Define a Fixture coordinate system—essentially an X, Y, and theta offset from the Pixel coordinate system. Any method that yields a point and angle is a potential source of a fixture. In a vision application, you commonly establish the fixture coordinate system in each image through a preliminary feature extraction step that finds a reliable landmark in the image. For this exercise, you manually enter the values **10** in cell **A1**, **20** in **A2**, and **45** in **A3**:

A2=			
	A	B	C
0	Image		
1	10.000	20.000	45.000
2			

3. From **A2**, open **Fixture** under **Coordinate Transforms**, and then open the TransFixtureToPixel property sheet.
4. On the property sheet, select the **Fixture** heading. By default, In-Sight assumes you want to select the fixture value by interactive reference. Move to cell **A1** and (holding down the O button) select the cells **A1**, **A2**, and **A3**:

Ref:	$A$1:$C$1		
	A	B	C
0	Image		
1	10.000	20.000	45.000
2	10.000		

5. Release O, and then click × to accept. In-Sight returns to the property sheet, which shows 10 as the fixture row, 20 as the fixture column, and 45 as the fixture angle.

6. From the property sheet, select the **Point** heading. Set the row value to 100 and the column value to 200.

7. Click **OK** to accept the fixture and point values. TransFixtureToPixel() returns a <u>Point structure</u> that stores the Pixel row and column values. As a convenience, it also emits the row and column values to the adjacent cells:

A2=	TransFixtureToPixel($A$1,$B$1,$C$1,100,200)			
	A	B	C	D
0	Image			
1	10.000	20.000	45.000	
2	Points	-60.711	232.132	

Other functions that depend on the translated point values can refer to the Point structure or the emitted values to obtain them.

8. To translate a point in Pixel coordinates to Fixture coordinates, open the <u>TransPixelToFixture</u> property sheet and specify a fixture and point through the techniques described above. TransPixelToFixture() similarly returns a Point structure and emits row and column values:

A3=	TransPixelToFixture($A$1,$B$1,$C$1,100,200)			
	A	B	C	D
0	Image			
1	10.000	20.000	45.000	
2	Points	-60.711	232.132	
3	Points	190.919	63.640	

COGNEX In-Sight

◄ ▲ ► Warp Transforms: Getting Started

A warped coordinate system is the result of mapping a curved region into a rectangular image space. With In-Sight, you can map a point from a warped image back to the equivalent point in the original unwarped image. This topic describes how to create a warped image and how to translate a point in the warped image back to the unwarped image.

► **To Warp an Image and Map a Point from It**

1. Acquire an appropriate image. For this exercise, a coin that has text printed around its perimeter makes a good sample image:
   a. Place the coin under the camera.
   b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image. The image should have good contrast without shadows, reflections, or other image defects.
   c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

   To acquire a new image, hold ☐ and click ✕.

2. Create a warped image of the coin, attempting to "straighten out" text printed around its perimeter:
   a. From cell **A1**, click ✕ to open the Formula Builder. From **Vision Processing**, open the **Image** heading, and then open the <u>WarpImage</u> property sheet.
   b. Click the **Region** heading. In-Sight draws the interactive region cursor on the image. Clicking ○ cycles the cursor to set location, size, angle, or curvature:
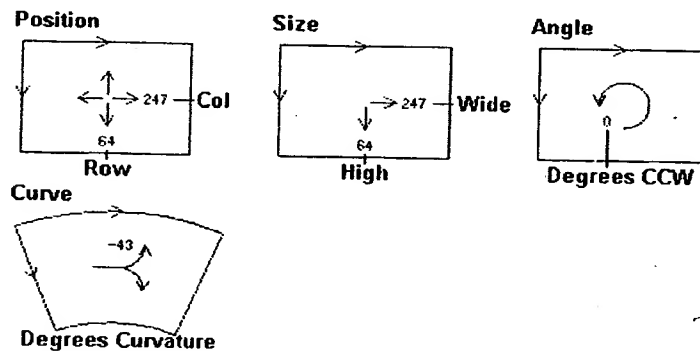
   c. Move the center of the Region Cursor to the center of the region; click ○ and resize the region; click ○ and set the angle (if any); and click ○ again and set the curvature. The result should look about like this:

   d. Click ✕ to accept the curved region. In-Sight returns to the property sheet.

e. Click **OK** to accept the input values. WarpImage() warps the curved region into a new rectangular image, stored in an <u>Image structure</u>. In-Sight displays the warped image whenever you fly over the Image structure containing it:

Any function that accepts an Image structure as an image source can refer to the warped image in lieu of the acquired image, simply by changing its Image reference from A0 to the cell containing the warped image.

3. Translate a point in the warped image back to equivalent point in the image:
   a. From cell A2, click ✕ to open the Formula Builder. From **Coordinate Transforms**, open the **Calibrate** heading, and then open the <u>TransWarpToPixel</u> property sheet.
   b. Click the **WarpImage** parameter. In-Sight switches to interactive reference mode. Move the <u>marquee</u> to A1 and click ✕, creating a reference to the warped image.
   c. Define a point in the warped image for translation. One method is to expand the **Point** heading in the TransWarpToPixel property sheet and directly edit the values. Another (shown below) is to put the values in cells and then refer to them.
   d. Click **OK** to accept the input values. TransWarpToPixel() translates the point from warped image back to the unwarped image, returning a Point structure and emitting its row and column values:

Acquired image — (pointing to spreadsheet)
Warped image — (pointing to spreadsheet)
Point in warped image — (pointing to spreadsheet)
Unwarped point (emitted results) — (pointing to spreadsheet)

COGNEX In-Sight

◄ ▲ ► Coordinate Transforms Reference

<u>Calib structure</u>   Stores a calibrated transformation. Created by Calibrate().

<u>Calibrate</u>   Creates a calibrated transformation between Pixel and World coordinates, given four points in both systems. Stores the transform in a Calib structure.

<u>TransBlobToWorld</u>   Applies a calibrated transformation to a Blobs structure. Returns a new Blobs structure translated to World coordinates.

<u>TransDistToWorld</u>   Applies a calibrated transformation to a Dist structure. Returns a new Dist structure translated to World coordinates.

Appendix I - Page **147**

TransEdgeToWorld	Applies a calibrated transformation to an Edges structure. Returns a new Edges structure translated to World coordinates.
TransFixtureToPixel	Converts a point in Fixture coordinates to Pixel coordinates. Returns a Point structure. Emits the Pixel coordinates.
TransPatternsToWorld	Applies a calibrated transformation to a Patterns structure. Returns a new Patterns structure translated to World coordinates.
TransPixelToFixture	Converts a point in Pixel coordinates to Fixture coordinates. Returns a Point structure. Emits the Fixture coordinates.
TransPixelToWorld	Converts a point in Pixel coordinates to World coordinates. Returns a Point structure. Emits the World coordinates.
TransWarpToPixel	Maps a point in a warped image back to the original, unwarped image. Returns a Point structure. Emits the Pixel coordinates.
TransWorldToPixel	Converts a point in World coordinates to Pixel coordinates. Returns a Point structure. Emits the Pixel coordinates.

COGNEX In·Sight

◄ ▲ ► Calib structure

Description	Stores a calibrated transformation. Created by Calibrate().
Values	Calib has no internal values that are read individually by Data Access functions. It stores a calibration on behalf of other functions requiring one.
Comments	The Calibrate function creates a calibrated transformation and stores it in a Calib structure. Functions that refer to a Calib structure to obtain calibration data are TransBlobToWorld, TransEdgeToWorld, TransPixelToWorld, TransPatternsToWorld, and TransWorldToPixel.
See Also	Calibrate

COGNEX In·Sight

◄ ▲ ► Calibrate function

Description	Creates a calibrated transformation between Pixel and World coordinates, given four points in both systems. Stores the transform in a Calib structure.	
Heading	Coordinate Transforms/Calibrate	
Syntax	Calibrate(Pixel Point 0, World Point 0, ..., Show)	
Inputs	**Pixel Point 0**	The image pixel corresponding to World Point 0. Row and column offsets from the image origin.
	—**Row**	
	—**Column**	
	World Point 0	The real-world point corresponding to Pixel Point 0. X and Y coordinates.
	—**X**	
	—**Y**	
	Pixel Point 1	The image pixel corresponding to World Point 1. Row and column offsets from the image origin.
	—**Row**	

—Column	
World Point 1	The real-world point corresponding to Pixel Point 1. X and Y
—X	coordinates.
—Y	
PixelPoint 2	The image pixel corresponding to World Point 2. Row and column
—Row	offsets from the image origin.
—Column	
World Point 2	The real-world point corresponding to Pixel Point 2. X and Y
—X	coordinates.
—Y	
Pixel Point 3	The image pixel corresponding to World Point 3. Row and column
—Row	offsets from the image origin.
—Column	
World Point 3	The real-world point corresponding to Pixel Point 3. X and Y
—X	coordinates.
—Y	
Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Calib**	On success, a Calib structure, which stores the calibrated transformation.
Emits	Nothing.	
Errors	Invalid world-to-pixel association. For example, the points are colinear.	
Comments	None.	
See Also	TransPixelToWorld, TransWorldToPixel, *Getting Started with Calibration*.	

COGNEX In Sight

◄ ▲ ► TransBlobToWorld function

Description	Applies a calibrated transformation to a Blobs structure. Returns a new Blobs structure translated to World coordinates.	
Heading	Coordinate Transforms/Calibrate	
Syntax	TransBlobToWorld(Calib, Blobs, Num to Convert)	
Inputs	**Calib**	Reference to a Calib structure, from Calibrate.
	Blobs	Reference to a Blobs structure, from ExtractBlobs, FindBlobs, or SortBlobs.
	Num to Convert	The number of blobs to convert.
Stores	**Blobs**	On success, a Blobs structure, with coordinates in World units.

Appendix I - Page **149**

Emits	Index	Zero-based index into the Blobs structure.
	Row	Centroid X coordinate, in World units. Read by GetRow.
	Col	Centroid Y coordinate, in World units. Read by GetCol.
Errors		Invalid parameter.
Comments		A Blobs structure stores a database of blobs extracted from an image. Each blob has a zero-based index and a set of measured values. When first created, many of these values are in image coordinates. TransBlobToWorld() converts all of the applicable values in one step, returning a new Blobs structure with coordinate values (Row, Column, ??and Angle??) in World units. Non-coordinate values (Area, Blob Color, Elongation, Holes, Number Found, Perimeter, Score, and Spread) pass through unchanged.
See Also		Blobs structure, Calib structure, Calibrate function, *Calibration: Converting Values*.

COGNEX In-Sight

◄ ▲ ► TransDistToWorld function

Description		Applies a calibrated transformation to a Dist structure. Returns a new Dist structure translated to World coordinates.
Heading		Coordinate Transforms/Calibrate
Syntax		TransDistToWorld(Calib, Dist)
Inputs	Calib	Reference to a Calib structure, from Calibrate.
	Dist	Reference to a Dist structure.
Stores	Dist	On success, a Dist structure, with coordinates in World units.
Emits	Index	Zero-based index into the Dist structure.
	Row0	First end point, X coordinate, in World units. Read by GetRow.
	Col0	First end point, Y coordinate, in World units. Read by GetCol.
	Row1	Second end point, X coordinate, in World units. Read by GetRow.
	Col1	Second end point, Y coordinate, in World units. Read by GetCol.
	Distance	The measured distance between the two end points. Read by GetDistance.
	Angle	Angle value, in counterclockwise degrees. Read by GetAngle.
Errors		Invalid parameter.
Comments		None.
See Also		Dist structure, Calib structure, Calibrate function, *Calibration: Converting Values*.

COGNEX In-Sight

◄ ▲ ► TransEdgeToWorld function

Description	Applies a calibrated transformation to an <u>Edges structure</u>. Returns a new Edges structure translated to World coordinates.	
Heading	Coordinate Transforms/Calibrate	
Syntax	TransEdgeToWorld(Calib, Edges, Num to Convert)	
Inputs	**Calib**	Reference to a <u>Calib structure</u>, from <u>Calibrate</u>.
	Edges	Reference to an Edges structure.
	Num to Convert	Number of edges to convert to World coordinates.
Stores	**Edges**	On success, an Edges structure, with coordinates in World units.
Emits	The exact values depend on the type of edges. For straight-line edges, the emitted values are:	
	Index	Zero-based index into the Edges structure.
	Row0	First end point, X coordinate, in World units. Read by <u>GetRow</u>.
	Col0	First end point, Y coordinate, in World units. Read by <u>GetCol</u>.
	Row1	Second end point, X coordinate, in World units. Read by <u>GetRow</u>.
	EndCol	Second end point, Y coordinate, in World units. Read by <u>GetCol</u>.
	Score	The edge score, 0.00 to 100.00. This value does not change because it is not a coordinate. Read by <u>GetScore</u>.
Errors	Invalid parameter.	
Comments	An Edges structure stores a database of edges extracted from an image. Each edge has a zero-based index and a set of measured values. When first created, many of these values are in image coordinates. TransEdgeToWorld() converts all of the applicable values (the row and column coordinates) in one step, returning a new Edges structure with coordinates in World units. Non-coordinate values (score) pass through unchanged.	
See Also	<u>Calib structure</u>, <u>Calibrate function</u>, <u>Edges structure</u>.	

COGNEX In Sight

◄ ▲ ► TransFixtureToPixel function

Description	Converts a point in Fixture coordinates to Pixel coordinates. Returns a Point structure. Emits the Pixel coordinates.	
Heading	Coordinate Transforms/Fixture	
Syntax	TransFixtureToPixel(Fixture, Point)	
Inputs	**Fixture**	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Point	A point, defined by X and Y offsets from the Fixture origin.
	—X	
	—Y	
Stores	**Point**	On success, a <u>Point structure</u>, which stores the points.

Emits	**Row**	Row coordinate, in Pixel units. A pixel offset from the image origin. Read by GetRow.
	Col	Column coordinate, in Pixel units. A pixel offset from the image origin. Read by GetCol.
Errors		Invalid parameter.
Comments		None.
See Also		TransPixelToFixture, *Getting Started with Fixture Transforms*.

COGNEX In·Sight

◀ ▲ ▶ TransPatternsToWorld function

Description		Applies a calibrated transformation to a Patterns structure. Returns a new Patterns structure translated to World coordinates.
Heading		Coordinate Transforms/Calibrate
Syntax		TransPatternsToWorld(Calib, Patterns, Num to Convert)
Inputs	**Calib**	Reference to a Calib structure, from Calibrate.
	Patterns	Reference to a Patterns structure.
	Num to Convert	The number of matched instances to convert.
Stores	**Patterns**	On success, a Patterns structure, with coordinates in World units.
Emits	**Index**	Zero-based index into the Patterns structure.
	Row	X coordinate of the reference point, in World units. Read by GetRow.
	Col	Y coordinate of the reference point, in World units. Read by GetCol.
	Angle	Angle of instance, in counterclockwise degrees relative to the model, converted to World units. Read by GetAngle.
	Scale	Scale of the instance, as a percentage of model. Not changed by the transform because it is not a coordinate. Read by GetScale.
	Score	Score of the instance. Not changed by the transform because it is not a coordinate. Read by GetScore.
Errors		Invalid parameter.
Comments		A Patterns structure stores a the matched instances extracted from an image. Each instance has a zero-based index and a set of measured values. When first created, many of these values are in image coordinates. TransPatternsToWorld() converts all of the applicable values (Row, Column, and Angle) in one step, returning a new Patterns structure with coordinates in World units. Non-coordinate values (Number Found, Scale, and Score) pass through unchanged.
See Also		Calib structure, Calibrate function, Patterns structure.

COGNEX In·Sight

◀ ▲ ▶ TransPixelToFixture function

Description	Converts a point in Pixel coordinates to Fixture coordinates. Returns a Point structure. Emits the Fixture coordinates.	
Heading	Coordinate Transforms/Fixture	
Syntax	TransPixelToFixture(Fixture, Point)	
Inputs	**Fixture**	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or both.
	—Column	**Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta	rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See _Working with Fixtures_.
	Point	A point, in Pixel coordinates (row and column offsets from the image
	—Row	origin).
	—Column	
Stores	**Point**	On success, a _Point structure_, which stores the points.
Emits	**Row**	X coordinate, in Fixture units. Read by _GetRow_.
	Col	Y coordinate, in Fixture units. Read by _GetCol_.
Errors	Invalid parameter.	
Comments	None.	
See Also	_TransFixtureToPixel_, _Getting Started with Fixture Transforms_.	

COGNEX In-Sight

◄ ▲ ► TransPixelToWorld function

Description	Converts a point in Pixel coordinates to World coordinates. Returns a Point structure. Emits the World coordinates.	
Heading	Coordinate Transforms/Calibrate	
Syntax	TransPixelToWorld(Calib, Point)	
Inputs	**Calib**	Reference to a _Calib structure_, from _Calibrate_.
	Point	A point, in Pixel coordinates (row and column offsets from the image
	—Row	origin).
	—Column	
Stores	**Point**	On success, a _Point structure_, which stores the points.
Emits	**Row**	X coordinate, in World units. Read by _GetRow_.
	Col	Y coordinate, in World units. Read by _GetCol_.
Errors	Invalid parameter.	
Comments	None.	
See Also	_Calibrate_, _TransPixelToWorld_, _TransWorldToPixel_	

COGNEX In-Sight

◄ ▲ ► TransWarpToPixel function

Description	Maps a point in a warped image back to the original, unwarped image. Returns a Point structure. Emits the Pixel coordinates.
Heading	Coordinate Transforms/Warp
Syntax	TransWarpToPixel(Image, Point, Show)

Inputs	**Image**	A warped image. A reference to an Image structure created by WarpImage. NOT the default image stored in A0.
	Point	A point in the warped image.
	—Row	
	—Column	
	Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Point**	On success, a Point structure, which stores the points.
Emits	**Row**	Row coordinate, in Pixel units. A pixel offset from the image origin. Read by GetRow.
	Col	Column coordinate, in Pixel units. A pixel offset from the image origin. Read by GetCol.
Errors	Invalid parameter.	
Comments	None.	
See Also	WarpImage, *Getting Started with Warp Transforms*.	

COGNEX In·Sight

◄ ▲ ► TransWorldToPixel function

Description	Converts a point in World coordinates to Pixel coordinates. Stores a Point structure. Emits the Pixel coordinates.
Heading	Coordinate Transforms/Calibrate
Syntax	TransWorldToPixel(Calib, Point)

Inputs	**Calib**	Reference to a Calib structure, from Calibrate.
	Point	A point, defined by X and Y offsets from the World origin.
	—X	
	—Y	
Stores	**Point**	On success, a Point structure, which stores the points.
Emits	**Row**	Row coordinate, in Pixel units. A pixel offset from the image origin. Read by GetRow.
	Col	Column coordinate, in Pixel units. A pixel offset from the image origin. Read by GetCol.
Errors	Invalid parameter.	
Comments	None.	

See Also   Calibrate, TransPixelToWorld

COGNEX In-Sight

◄ ▲ ► Edge: Overview

Edge detection is a <u>feature extraction</u> technique for finding single edges or edge pairs in an image. Each edge is marked by a change in the grayscale value of neighboring pixels from light-to-dark or dark-to-light. This change might span several pixels.

This section describes In-Sight's Edge functions, located under <u>Vision Processing</u>, which offer your application data about the number, location, shape, and orientation of edges. It discusses the following topics:

- <u>Getting Started with Edge Detection</u>
- <u>Understanding Edge Detection</u>
- <u>Edge Reference</u>.

In-Sight's Edge functions offer methods of filtering false (noise-induced) edges and excluding uninteresting edges. They also provide methods for sorting them in various orders.

COGNEX In-Sight

◄ ▲ ► Edge: Getting Started

This topic introduces edge detection. It uses the FindLine() function, which detects straight-line edges, as an example to show the basic procedures. Other Edge functions detect curved, circular, or multiple edges using through similar sequences of steps.

► To Detect an Edge

1. Acquire an appropriate image. For this exercise:
   a. Open the <u>part sample image</u>, print it, and return to this topic. Place the printed image under the camera.
   b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode.* Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
   c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

   To acquire a new image, hold ☐ and click ✕.

2. From an empty cell, click ✕ to open the <u>Formula Builder</u>.
   NOTE Try to leave a blank row above functions that emit formulas. In-Sight prints descriptive labels above emitted formulas if it has room for the labels. If the row above is occupied, it emits only the numeric values, without labels.
3. From the **Vision Processing** category, select **Edge**, and then open the <u>FindLine</u> property sheet.
4. To define region of interest, select the Region head and click ✕. In-Sight hides the worksheet and

displays the Region cursor, which you interactively <u>adjust</u> with the Control Pad. When detecting edges—which can be black-to-white or white-to-black transitions—In-Sight indicates the region's scanning direction with arrowheads along two edges:
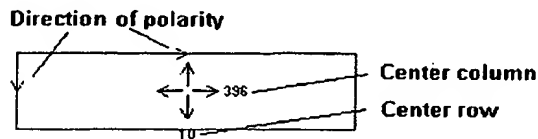
In the orientation shown, FindLine() interprets edge polarity from left-to-right or from top-to-bottom.

5. Outline an image area containing one or more edges, and then click ✕ to accept the region, returning to the FindLine() property sheet. At the bottom of the property sheet, In-Sight draws an edge-strength graph on the image:
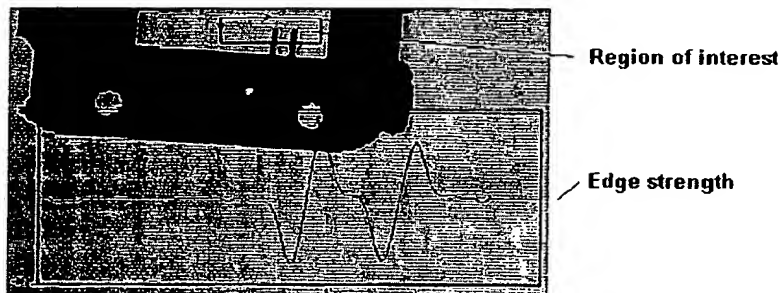
To view only the edge-strengh graph and image, click Δ to toggle the property sheet overlay ON or OFF.

6. In the FindLine() property sheet, set the parameters that define edge characteristics:
   o **Polarity**, either white-to-black or black-to-white.
   o **Find By**, xxx
   o **Accept Threshold**, the minimum acceptable edge score. Linear features scoring below this level are rejected.
   o **Angle Range**,
   o **Edge Width**, the minimum edge width, in pixels.

7. After you set appropriate parameters, either:
   o Click **Run** to extract edges without closing the property sheet.
   o Click **OK**, to extract edges and close the property sheet.

FindLine() analyses the image for edges that fit the selection criteria, stores an Edges structure containing the edges, and emits various results:

1	11.000	Row0	Col0	Row1	Col1	Score
2	Edges	152.500	320.841	181.500	320.841	–10.949

Each Edge has a zero-based index and a set of measured values. You use the indexes to refer to a particular edge in the Edges structure.

Appendix I - Page 157

To use a result value in a formula, simply create a reference to its cell. If you don't need one of the automatically generated result cells, you can <u>delete it</u>.

8. You can experiment with different selection criteria by changing their values and clicking **Run** again. Restrictive values tend to yield fewer edges, and relaxed values tend to yield more edges.
9. By default, In-Sight displays the edge graphics only when you highlight the Edges structure, as described in <u>*Flyover Graphics*</u>. To display the Edge graphics all the time, set the FindLine() function's **Show** parameter to **Show All**.

COGNEX In-Sight

◄ ▲ ► Edge: Understanding

Edge analyzes images in two stages:

1. *Projection*, which maps a two-dimensional window of an image (the *edge window*) into a one-dimensional image. Projection collapses an image by summing the pixels in each line along the direction of the projection, and then taking the first derivative of the line represented by these sums. This technique tends to amplify edges along the axis of the projection.
2. *Filtering*, which subjects an image projection to constraints that eliminate the minor changes in grayscale values likely to be the result of image noise. After filtering, the results that remain significant are considered closest to the true edges in the image.

The following topics tell you how use edge

- <u>Defining an ROI</u>
- <u>Specifying Edge Features</u>
- <u>Specifiying Edge Filter Parameters</u>
- <u>Setting the Acceptance Threshold</u>
- <u>Specifying the Extract Method</u>
- <u>Specifying the Sort Order</u>
- <u>Finding Edges</u>
- <u>Measuring Edges</u>.

COGNEX In-Sight

◄ ▲ ► Edge: Defining the ROI

When you open the Region heading from an Edge function, In-Sight draws a rectangular outline called the *region cursor* over the image. The region cursor indicates region of interest and lets you move and rotate it

with the Control Pad, as described in *Defining a Region of Interest.*

An edge is a transition from black-to-white or white-to-black—an attribute called *polarity*. When defining the properties of an edge, you can specify the polarity to detect. Transitions that don't have the right polarity aren't edges. Polarity depends on direction, however. An edge that is white-to-black in one direction is black-to-white in the opposite direction. The region of interest therefore has a detection direction, indicated by arrowheads on two sides of the region cursor:
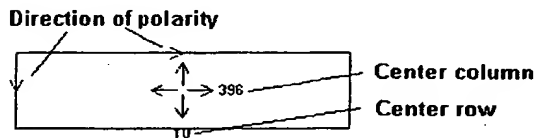
Ideally, the x-axis of the region cursor should be perpendicular to the edges you expect to find. Assuming the base point is at the upper left of the region cursor, the search for edges starts at the left side of the region and progresses to the right side. Edge finds all edges perpendicular to the region cursor's angle of rotation, subject to the search criteria you have selected:
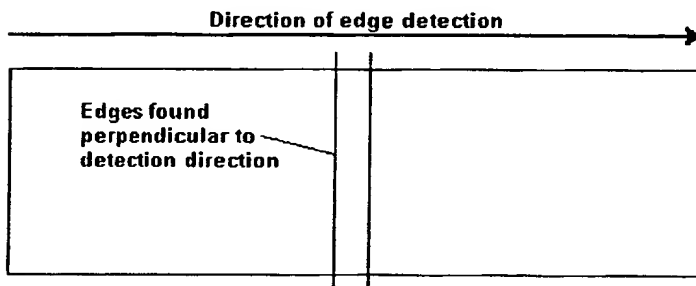
This relationship is the same regardless of how the region cursor is rotated.

COGNEX In-Sight

◄ ▲ ► Edge: Specifying Features

The Feature section, in the upper-left corner of the Edge dialog, contains controls that determine the type of edges sought. Its has four fields: FirstPolarity, ExpectedWidth, Tolerance, and Second Polarity.

FirstPolarity sets the expected grayscale progression for edges: dark-to-light, light-to-dark, or either. By default, FirstPolarity is zero, indicating "either."

ExpectedWidth sets the number of pixels expected between edge pairs. It is also zero by default—which configures Edge to seek single edges, not edge pairs. Changing ExpectedWidth to a positive value other than zero configures Edge to seek edge pairs separated by the specified number of pixels.

Setting ExpectedWidth enables the Tolerance field, where you specify the number of pixels of deviation from the ExpectedWidth value. It also enables the Second Polarity list, where you specify a polarity for the last edge in the pair.

Second Polarity includes an option (in addition to the ones in FirstPolarity) called SolidBar. SolidBar, when set, ensures that edge pairs enclose an area that does not change color. For example, if the leading edge in such a pair is light-to-dark, the trailing edge will be the first dark-to-light progression afterward along the edge detection direction. The resulting pair would then enclose a solid (in this case, dark) bar.
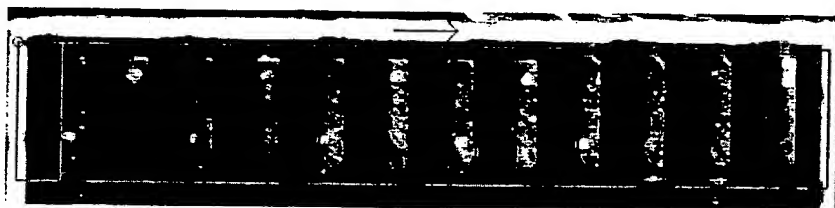
COGNEX In-Sight

◄ ▲ ► Edge: Filtering Edges

After projection, Edge applies an edge filter to the resulting one-dimensional image. The edge filter accentuates edges in the image and produces a filtered image. The Operator section lets you specify two filtering parameters, Noise Size and Leniency.

Size (or Noise Size) determines the number of adjacent points considered in the derivation of the one-dimensional representation. For each point in the line, the derivation includes the specified number of adjacent points on *each side* of the point. The higher the setting, the higher the contrast required in the image to avoid being "smoothed out" in the derivation.

Leniency specifies the number of pixels to ignore at an edge in the image. Due to slight changes in edge or edge-pair position or orientation, a pixel on the edge of a feature might not always have the same grayscale value in each acquired image at run time. Setting a leniency zone between the light and dark areas of a feature minimizes the effects of slight changes in feature location on edge filter results.

COGNEX In-Sight

◄ ▲ ► Edge: Accept and Num to Find

The Accept value specifies the *acceptance threshold*—the lowest score that is a valid edge. Features with

lower scores are rejected as edges. Scoring depends on the grayscale difference between two adjacent edges. These grayscale values are normalized so that 0 is black and 100 is white. The contrast score is the difference between them. Therefore, the highest possible score is 100, and the lowest is 0. When you set the Accept threshold, you are determining the largest grayscale difference that may be discarded as a possible edge.

N2Find allows you to set the maximum number of edges to find. ?? immediately stops looking for edges after it finds N2Find edges that exceed the acceptance threshold. It might or might not find the specified number.

COGNEX In-Sight

◄ ▲ ► Edge: Extract Method

After you've specified the appropriate filtering operator, you can specify which edges to find when you click Find or Run(Find). The Extract menu gives you a choice between accepting the *first* edges found, halting evaluation as soon as it reaches N2Find, or the *best* edges found, evaluating the entire window and then selecting the highest scores. In either case, all results must pass the Accept threshold to be reported as edges.

By default, if you specify a smaller number of edges in N2Find than actually exist in the image, the ??? function will report a number edges equal to N2Find in the order they appear in the edge detection direction. In this case, the result is already sorted by Offset. If you choose to extract by score, all of the edges found over the entire width of the region will be evaluated. If the number of edges found exceeds N2Find then only the best N2Find edges are reported. In this case, the order of the results is undefined.

COGNEX In-Sight

◄ ▲ ► Edge: Specifying Sort Order

Before your object has found edges, you can arrange to sort them by any of the values mentioned in the report. To sort the edges in the order you desire, choose your method from the Sort list. Later, when you instruct the object to display the edges it found, they will appear sorted in the order you specified.

The Sort list lets you specify the criterion for sorting the edges:

- **0**: Unsorted.
- **1**: Offset distance, in pixels, from the focal point.
- **2** : Scores (contrast values).
- **3**: X positions, in pixels, relative to the origin point.
- **4**: Y positions, in pixels, relative to the origin point.

- 5: Widths, in pixels.
- 6: Delta widths—that is, the change in width relative to the previous edge's width.
- 7: Positive polarity first, negative polarity last.
- 8: Negative polarity first, positive polarity last.

COGNEX In·Sight

◄ ▲ ► Edge: Finding Edges

Find first identifies edges, according to the one-dimensional, derived, filtered image. It next sorts the found edges according to the Sort selection and reports the results in the Result area in the bottom of the dialog box. Finally, it reports the number of edges found in the NFound field and their locations, scores, and other results in the Result list:

The Result for each edge found contains a series of values, representing the grid coordinates of the edge in relation to the region and to the image at large. These values are described below, as they appear from left to right:

- **Index**: A number by which to refer to any particular found edge among the others.
- **X**: The x-axis coordinate of the found edge, measured in pixels from the origin of the entire image.
- **Y**: The y axis coordinate of the found edge, measured in pixels from the origin of the entire image.
- **Offset**: The offset (in pixels) from the focal point of the scan region.
- **Width**: The width in pixels of the edge pair, if applicable.
- **Score**: The difference in grayscale at the edge, measured in incremental shades of gray from 0.00 to 100.00.
- **P1**: The polarity of the first edge, expressed as –1 (light to dark progression), 1 (dark to light progression) or 0 (either progression).
- **P2**: The polarity of the second edge, if applicable.

COGNEX In·Sight

◄ ▲ ► Edge: Measuring Edges

???An additional button in the Edge structure dialog, World, allows measurements for offset, x coordinate, y coordinate, and width to show in either pixels or real-world units of measurement. If you click on World to

choose world measurements, the button changes to "Pixel" to toggle back to pixels, and vice versa. Use a calibrated transformation to convert to world measurements as necessary.

═══════════════════════════════ **COGNEX In Sight**

◄ ▲ ► Edge: Function Reference

Edges structure	Stores a database of edges, each with a zero-based index and a profile of measured values. Coordinates in Image units (pixels) unless converted by TransEdgeToWorld.
FindCircle	Finds the best circular edge. Returns an Edges structure. Emits the center row, center column, and radius.
FindCurve	Finds the best curved edge. Returns an Edges structure.
FindLine	Finds the best straight-line edge. Returns an Edges structure.
FindMultiLine	Finds multiple straight-line edges. Returns an Edges structure.
FindSegment	Finds an edge pair defined by a black or white segment. Returns an Edges structure. Emits the two edges.
PairDistance	Returns the distance between a pair of edges.
PairEdges	Groups multiple edge results into pairs. Returns an Edges structure.
PairMaxDistance	Returns the maximum distance between multiple edge pairs.
PairMeanDistance	Returns the mean distance between multiple edge pairs.
PairMinDistance	Returns the minimum distance between multiple edge pairs.
PairSDevDistance	Returns the standard deviation of the distances multiple edge pairs..
SortEdges	Sorts an Edges structure by specified criteria. Stores a new, sorted Edges structure.

TransEdgeToWorld, under Coordinate Transforms, converts edge results to World coordinates. Results functions, under Data Access, that get results from an Edges structure are GetCol, GetNFound, GetRadius, GetRow, and GetScore.

═══════════════════════════════ **COGNEX In Sight**

◄ ▲ ► Edge: Edges structure

Description	Stores a database of edges, each with a zero-based index and a profile of measured values. Coordinates in Image units (pixels) unless converted by TransEdgeToWorld.

Values

Value	Description	Access function
Index	Index identifiying a particular edge. Zero-based, so the first edge is indexed 0.00, the second is indexed 1.00, and so on.	—
Column	???. In pixels unless converted by TransEdgeToWorld().	GetCol
NFound	Number of edges ?or edge pairs? stored.	GetNFound
Radius	???. ??Always in pixels, even if the Edges structure is converted to World units.	GetRadius

Row	???. In pixels unless converted by TransEdgeToWorld().	GetRow
Score	As computed by ??, 0.00 to 100.00.	GetScore

Comments   None.
See Also   FindCircle, FindCurve, FindLine, FindMultiLine, PairEdges, FindSegment, TransEdgeToWorld, SortEdges.

COGNEX In Sight

◄ ▲ ► Edge: FindCircle function

Description Finds the best circular edge, returning an Edges structure. Emits the center row, center column, and radius.

Heading Vision Processing/Edge

Syntax FindCircle(Image, Fixture, Torus, ..., Show)

Inputs

Image	Image data source, a reference to an Image structure. Default is A0.	
Fixture	Fixture origin, defined by row, column, and angle offsets, typically to	
—Row	compensate for image-to-image variation in position, orientation, or	
—Column	both. **Row** and **Column** are offsets, in pixels from the image origin.	
—Theta	**Theta** is a rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.	
Torus	Region of interest. **X** and **Y** define the center of the torus by offsets	
—X	from the Fixture origin. **Inner radius** is the radius of the inner circle, in	
—Y	pixels. **Outer radius** is the radius of the outer circle, in pixels.	
—Inner		
—Outer		
Polarity	Expected grayscale transition, evaluated from the inside to the outside. One of: either, black-to-white, or white-to-black	
Find By	Criterion for selecting circular edges. One of:	

best score
smallest circle
largest circle.

Accept Thresh Acceptance threshold, a minimum score, 0 through 100.
Edge Width Expected transition width, in pixels.
Show Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores **Edges** On success, an Edges structure, which stores the edge results. ??Should store a Circle, not an Edges structure??

Emits	centRow	The row coordinate of the center point. Read by GetRow.
	centCol	The column coordinate of the center point. Read by GetCol.
	radius	The radius of curvature, in pixels. Read by GetRadius.
	score	The edge score, 0.00 to 100.00, a floating-point value. Read by GetScore.
Errors	Invalid parameter.	
Comments	None.	
See Also	FindCurve, FindLine, FindMultiLine, PairEdges, PairDistance, PairMaxDistance, PairMeanDistance, PairMinDistance, PairSDevDistance, FindSegment, SortEdges, TransEdgeToWorld	

COGNEX In·Sight

◄ ▲ ► Edge: FindCurve function

Description	Finds the best curved edge, returning an Edges structure. ??JM thinks this should not return an Edges. Suggested TBD.
Heading	Vision Processing/Edge
Syntax	FindCurve(Image, Fixture, Region, ..., Show)
Inputs	**Image**	Image data source, a reference to an Image structure. Default is A0.
	Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or
	—Column	both. **Row** and **Column** are offsets, in pixels from the image origin.
	—Theta	**Theta** is a rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Region	Region of interest. **X** and **Y** define the top-left point by offsets from the
	—X	Fixture origin, in pixels. **High** and **Wide** are the height and width, in
	—Y	pixels. **Angle** is a rotation from the Fixture X axis, ± 360° CCW.
	—High	**Curve** is the angle between the sides of the region, ± 360° CCW). For
	—Wide	curved regions, "width" is the arc distance at the center of the region.
	—Angle	See *Defining a Region of Interest*.
	—Curve	
	Polarity	Expected grayscale transition, evaluated from the inside to the outside. One of: either, black-to-white, or white-to-black.
	Find By	Criterion for selecting circular edges. One of:

- best score
- closest to Y axis (the top-most, unless the Region is "upside down")
- farthest from Y axis (the bottom-most, unless the Region is "upside down")

Accept Thresh	Acceptance threshold, a minimum score, 0 through 100.
Angle Range	Angular tolerance, ± 0 to 10 degrees.
Edge Width	Expected transition width, in pixels.
Show	Selects the graphics to display, as described in *Flyover Graphics:*

— ....—... .. .. . .p...., ...

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Emits	centRow	The row coordinate of Point0, the center point. Read by GetRow.
	CentCol	The column coordinate of Point0, the center point. Read by GetCol.
	StartRow	The row coordinate of Point1, the first point. Read by GetRow.
	StartCol	The column coordinate o. Point1, the first point. Read by GetCol.
	EndRow	The row coordinate of Point2, the end point. Read by GetRow.
	EndCol	The column coordinate of Point2, the end point. Read by GetCol.
	Radius	The radius of curvature, in pixels. Read by GetRadius.
	Score	The edge score, 0.00 to 100.00, a floating-point value. Read by GetScore.
Stores	Edges	On success, an Edges structure, which stores the edge results.
Errors	Invalid parameter.	
Comments	None.	
See Also	FindCircle, FindLine, FindMultiLine, PairEdges, PairDistance, PairMaxDistance, PairMeanDistance, PairMinDistance, PairsSDevDistance, FindSegment, SortEdges, TransEdgeToWorld	

COGNEX In Sight

◄ ▲ ► Edge: FindLine function

Description	Finds the best straight-line edge, returning an Edges structure.	
Heading	Vision Processing/Edge	
Syntax	FindLine(Image, Fixture, Region, ..., Show)	
Inputs	Image	Image data source, a reference to an Image structure. Default is A0.
	Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or
	—Column	both. Row and Column are offsets, in pixels from the image origin.
	—Theta	Theta is a rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See Working with Fixtures.
	Region	Region of interest. X and Y define the top-left point by offsets from the
	—X	Fixture origin, in pixels. High and Wide are the height and width, in
	—Y	pixels. Angle is a rotation from the Fixture X axis, ± 360° CCW.
	—High	Curve is the angle between the sides of the region, ± 360° CCW). For
	—Wide	curved regions, "width" is the arc distance at the center of the region.
	—Angle	See Defining a Region of Interest.
	—Curve	
	Polarity	Expected grayscale transition, evaluated from the inside to the outside. One of: either, black-to-white, or white-to-black.
	Find By	Criterion for selecting edges. One of:

- best score
- first edge
- last edge

	Accept Thresh	Acceptance threshold, a minimum score, 0 through 100.
	Angle Range	Angular tolerance, ± 0 to 10 degrees.
	Edge Width	Expected transition width, in pixels.
	Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Emits	**Number Found**	Number of edges found. Read by GetNFound.
	StartRow	The row coordinate of Point0, the first point. Read by GetRow.
	StartCol	The column coordinate of Point0, the first point. Read by GetCol.
	EndRow	The row coordinate of Point1, the end point. Read by GetRow.
	EndCol	The column coordinate of Point1, the end point. Read by GetCol.
	Score	The edge score, 0.00 to 100.00, a floating-point value. Read by GetScore.
Stores	**Edges**	On success, an Edges structure, which stores the edge results.
Comments	None.	
Errors	Invalid parameter.	
See Also	FindCircle, FindCurve, FindMultiLine, PairEdges, PairDistance, PairMaxDistance, PairMeanDistance, PairMinDistance, PairSDevDistance, FindSegment, SortEdges, TransEdgeToWorld, *Getting Started with Edge Detection*.	

COGNEX In·Sight

◄ ▲ ► Edge: FindMultiLine function

Description	Finds multiple straight-line edges, returning an Edges structure.	
Heading	Vision Processing/Edge	
Syntax	FindMultiLine(Image, Fixture, Region, ..., Show)	
Inputs	**Image**	Image data source, a reference to an Image structure. Default is A0.
	Fixture	Fixture origin, defined by row, column, and angle offsets, typically to
	—Row	compensate for image-to-image variation in position, orientation, or
	—Column	both. **Row** and **Column** are offsets, in pixels from the image origin.
	—Theta	**Theta** is a rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures*.
	Region	Region of interest. **X** and **Y** define the top-left point by offsets from the
	—X	Fixture origin, in pixels. **High** and **Wide** are the height and width, in
	—Y	pixels. **Angle** is a rotation from the Fixture X axis, ± 360° CCW.
	—High	**Curve** is the angle between the sides of the region, ± 360° CCW). For

Appendix I - Page **167**

—Wide		curved reg ons, "w dtn" s tne arc d stance at tne center or tne reg on.
—Angle		See *Defining a Region of Interest*.
—Curve		
Number to Find		Maximum number of edges to report.
Threshold: B2W		Minimum, black-to-white edge, 0 through 100.
Threshold: W2B		Minimum, white-to-black edge, 0 through 100.
Find By		Criterion for selecting edges. One of:

- best score
- first edge
- last edge

Angle Range		Angular tolerance, ± 0 to 10 degrees.
Edge Width		Expected transition width, in pixels.
Show		Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	Edges	On success, an Edges structure, which stores the edge results.
Emits	index	The edge number, 0.00 through **n.00.**
	number found	Number of edges found. Read by GetNFound.
	startRow	The row coordinate of Point0, the first point. Read by GetRow.
	startCol	The column coordinate of Point0, the first point. Read by GetCol.
	endRow	The row coordinate of Point1, the end point. Read by GetRow.
	endCol	The column coordinate of Point1, the end point. Read by GetCol.
	score	The edge score, 0.00 to 100.00, a floating-point value. Read by GetScore.
Errors		Invalid parameter.
Comments		Extracts and stores points for as many edge intersection that fall within control limits up to the maximum number expected. The output is always sorted by offset from the X-axis origin.
See Also		FindCircle, FindCurve, FindLine, PairEdges, PairDistance, PairMaxDistance, PairMeanDistance, PairMinDistance, PairSDevDistance, FindSegment, SortEdges, TransEdgeToWorld

COGNEX In Sight

◄ ▲ ► Edge: FindSegment function

Description	Finds an edge pair defined by a black or white segment. Returns an Edges structure. Emits the two edges.	
Heading	Vision Processing/Edge	
Syntax	FindSegment(Image, Fixture, Region, ..., Show)	
Inputs	**Image**	Image data source, a reference to an _Image structure_. Default is A0.
	Region	Region of interest. **X** and **Y** define the top-left point by offsets from the
	—X	Fixture origin, in pixels. **High** and **Wide** are the height and width, in
	—Y	pixels. **Angle** is a rotation from the Fixture X axis, ± 360° CCW.
	—High	**Curve** is the angle between the sides of the region, ± 360° CCW). For
	—Wide	curved regions, "width" is the arc distance at the center of the region.
	—Angle	See _Defining a Region of Interest_.
	—Curve	
	Segment Color	One of black or white.
	Find By	Criterion for selecting edges. One of:
		widest segment
		strongest edges
	Accept Thresh	Acceptance threshold, a minimum score, 0 through 100.
	Angle Range	Angular tolerance, ± 0 to 10 degrees.
	Edge Width	Expected transition width, in pixels.
	Show	Selects the graphics to display, as described in _Flyover Graphics_:
		• Hide all
		• Result graphics only
		• Input and result graphics
		• Show all, which displays input, result, and chart (if any) graphics.
Stores	**Edges**	On success, an _Edges structure_, which stores the edge results.
Emits	**Index**	The edge number, 0.00 through **n**.00.
	Number Found	Number of edges found. Read by _GetNFound_.
	StartRow	The row coordinate of Point0, the first point. Read by _GetRow_.
	StartCol	The column coordinate of Point0, the first point. Read by _GetCol_.
	EndRow	The row coordinate of Point1, the end point. Read by _GetRow_.
	EndCol	The column coordinate of Point1, the end point. Read by _GetCol_.
	Score	The edge score, 0.00 to 100.00, a floating-point value. Read by _GetScore_.
Errors	Invalid parameter.	
Comments	None	
See Also	FindCircle, FindCurve, FindLine, FindMultiLine, PairEdges, PairDistance, PairMaxDistance, PairMeanDistance, PairMinDistance, PairSDevDistance, SortEdges, TransEdgeToWorld	

COGNEX In-Sight

Appendix I - Page **169**

◄ ▲ ► Edge: PairDistance function

Description	Returns the distance between a pair of edges.
Heading	Vision Processing/Edge
Syntax	PairDistance(Edges, First Edge, Second Edge)
Inputs	**Edges** Input edges, a reference to Edges structure.
	First Edge First edge in a pair.
	Second Edge Second edge in a pair.
Returns	Distance between a pair of edges. In pixels unless the input Edges structure has been converted to other units by TransEdgeToWorld.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	FindCircle, FindCurve, FindLine, FindMultiLine, PairEdges, PairMaxDistance, PairMeanDistance, PairMinDistance, PairSDevDistance, FindSegment, SortEdges, TransEdgeToWorld

COGNEX In-Sight

◄ ▲ ► Edge: PairEdges function

Description	Groups multiple edge results into pairs, returning an Edges structure.
Heading	Vision Processing/Edge
Syntax	PairEdges(Edges, Num of Pairs, ..., Show)
Inputs	**Edges** Input edges, a reference to an Edges structure.
	Num of Pairs Number of edges to compute.
	First Edge The polarity of the first edge. One of black-to-white, white-to-black, or either.
	Second Edge The polarity of the second edge. One of black-to-white, white-to-black, or either.
	Edge Dist Minimum and maximum distances between two edges, in pixels.
	—**Min**
	—**Max**
	Intermediate Edges Maximum and minimum numbers of intermediate white-to-black and black-to-white edges.
	—**Min B2W**
	—**Max B2W**
	—**Min W2B**
	—**Max W2B**
Stores	**Edges** An Edges structure, which stores the edge results.

Emits	Index	The edge pair number, 0.00 through n.00.
	Number Found	Number of edge pairs found. Read by GetNFound.
	StartRow	The row coordinate of Point0, the first point. Read by GetRow.
	StartCol	The column coordinate of Point0, the first point. Read by GetCol.
	EndRow	The row coordinate of Point1, the end point. Read by GetRow.
	EndCol	The column coordinate of Point1, the end point. Read by GetCol.
	Score	The edge score, 0.00 to 100.00, a floating-point value. Read by GetScore.
Errors	Invalid parameter.	
Comments	None.	
See Also	FindCircle, FindCurve, FindLine, FindMultiLine, PairDistance, PairMaxDistance, PairMeanDistance, PairMinDistance, PairSDevDistance, FindSegment, SortEdges, TransEdgeToWorld	

COGNEX In-Sight

◄ ▲ ► Edge: PairMaxDistance function

Description	Returns the maximum distance between multiple edge pairs.
Heading	Vision Processing/Edge
Syntax	PairMaxDistance(Edges)
Inputs	**Edges** Input edges, a reference to an Edges structure.
Returns	Minimum distance between points. In pixels unless the input Edges structure has been converted to other units by TransEdgeToWorld.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	FindCircle, FindCurve, FindLine, FindMultiLine, PairEdges, PairDistance, PairMeanDistance, PairMinDistance, PairSDevDistance, FindSegment, SortEdges, TransEdgeToWorld

COGNEX In-Sight

◄ ▲ ► Edge: PairMeanDistance function

Description	Returns the mean distance between multiple edge pairs.
Heading	Vision Processing/Edge
Syntax	PairMeanDistance(Edges)
Inputs	**Edges** Input edges, a reference to Edges structure.
Returns	Mean (average) distance between edge pairs. In pixels unless the input Edges structure has been converted to other units by TransEdgeToWorld.
Emits	Nothing

Errors	Invalid parameter.
Comments	None.
See Also	FindCircle, FindCurve, FindLine, FindMultiLine, PairEdges, PairDistance, PairMaxDistance, PairMinDistance, PairSDevDistance, FindSegment, SortEdges, TransEdgeToWorld

COGNEX In-Sight

◄ ▲ ► Edge: PairMinDistance function

Description	Returns the minimum distance multiple edge pairs.
Heading	Vision Processing/Edge
Syntax	PairMinDistance(Edges)
Inputs	**Edges** Edge data. A reference to an Edges structure.
Returns	Minimum distance between points. In pixels unless the input Edges structure has been converted to other units by TransEdgeToWorld.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	FindCircle, FindCurve, FindLine, FindMultiLine, PairEdges, PairDistance, PairMaxDistance, PairMeanDistance, PairSDevDistance, FindSegment, SortEdges, TransEdgeToWorld

COGNEX In-Sight

◄ ▲ ► Edge: PairSDevDistance function

Description	Returns the standard deviation of the distances between multiple edge pairs..
Heading	Vision Processing/Edge
Syntax	PairSDevDistance(Edges)
Inputs	**Edges** Input edges, a reference to an Edges structure.
Returns	Standard deviation of distance between point. In pixels unless the input Edges structure has been converted to other units by TransEdgeToWorld.
Emits	Nothing.
Errors	Invalid parameter.
Comments	None.
See Also	FindCircle, FindCurve, FindLine, FindMultiLine, PairEdges, PairDistance, PairMaxDistance, PairMeanDistance, PairMinDistance, FindSegment, SortEdges, TransEdgeToWorld

COGNEX In-Sight

◄ ▲ ► Edge: SortEdges function

Description	Sorts an Edges structure by specified criteria. Stores a new, sorted Edges structure.	
Heading	Vision Processing/Edge	
Syntax	SortEdges(Edges, Number to Sort, Sort By, Show)	
Inputs	**Edges**	Input edges, a reference to an Edges structure.
	Num to Sort	Number of edges to sort
	Sort By	The criterion for sorting. One of:

- best score
- first to last
- B2W by score, then W2B by score
- B2W first-to-last, then W2B first-to-last.

	Show	Show or hide the graphic.
Stores	**Edges**	On success, an Edges structure, which stores the edge results.
Emits	**Index**	The edge number, 0.00 through n.00.
	Number Found	Number of edges found. Read by GetNFound.
	StartRow	The row coordinate of Point0, the first point. Read by GetRow.
	StartCol	The column coordinate of Point0, the first point. Read by GetCol.
	EndRow	The row coordinate of Point1, the end point. Read by GetRow.
	EndCol	The column coordinate of Point1, the end point. Read by GetCol.
	Score	The edge score, 0.00 to 100.00, a floating-point value. Read by GetScore.
Errors	Invalid parameter.	
Comments	None.	
See Also	FindCircle, FindCurve, FindLine, FindMultiLine, PairEdges, PairDistance, PairMaxDistance, PairMeanDistance, PairMinDistance, PairSDevDistance, FindSegment, TransEdgeToWorld	

COGNEX In-Sight

◄ ▲ ► Geometry: Overview

The Geometry functions construct geometric shapes (under **Fit**) and gauge distances and angles (under **Measure**). Using them, you can find the distances, intersections, and angles of points, lines, and circles in various combinations. These geometric measurements are a crucial part of many inspection applications, which use them to confirm the dimensions and location of a part.

This section introduces the Geometry functions, discussing the following topics:

- Getting Started with Measurement
- Geometry Function Reference.

Most of the measurement functions return a Dist structure, which stores a line segment described by its end points, distance (its length), and an angle (from the image row axis, by default). They also emit formulas to place the current Dist values in a block of cells. If your application does not use one of the emitted formulas, you can remove it from the worksheet.

NOTE The measurement functions are independent from the unit of measure. They perform an appropriate computation on their floating-point inputs but don't "know" what the inputs represent. If the inputs are in Image units (rows and columns, in pixels), then the outputs are in Image units. But if the inputs are in World units (translated from the image through Calibrate), then the outputs are in World units.

COGNEX In Sight

◄ ▲ ► Geometry: Getting Started

In this topic, you measure the distance between two points with the PointToPoint function. The result looks like this:

NSDOCID: <WO___0111445A2_I_>

Completing this exercise gives you hands-on experience with geometric measurement. The steps listed here generalize to the other measurement functions.

▶ **To Measure the Distance between Two Points**

1. Acquire an appropriate image. For this exercise:
    a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
    b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
    c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

    To acquire a new image at any time, hold □ and click ✕.

2. Before you can measure a point-to-point distance, you need two points. For this exercise, ExtractBlobs is a convenient source of point data:
    a. Leaving cell **A1** empty, move to cell **A2** and click ✕ to open the Formula Builder.
        NOTE   Try to leave a blank row above functions that emit formulas. In-Sight prints descriptive labels above emitted formulas if it has room for the labels. If the row above is occupied, it emits only the numeric values, without labels.
    b. From **Vision Processing**, open the **Blob** heading, and then open the ExtractBlobs() property sheet.
    c. To define the region of interest, click the **Region** heading. In-Sight hides the worksheet and displays the Region cursor. Using the Control Pad, surround the part image, which has at most six blobs (four small round holes, one larger rectangular hole, and the entire perimeter). Click ✕ to return to the property sheet.
    d. Increase the **Number to Find** value to **2** and click ✕ to accept it. (This value determines the number of blobs to extract. Each blob has a centroid point, and the two centroids define a distance to be measured.)
    e. Decrease the **Area Limit: Max** value to **1000** (to exclude the perimeter and the rectangular hole as blobs), click ✕ to accept, and click **OK** to close the property sheet. ExtractBlobs() extracts two blobs, stores them in a Blobs structure, and emits various formulas:

The values of interest here are the centroid Row and Column values. You can ignore or remove

the other emitted values.
3. Measure the distance between the points:

a. From cell **A5**, click ✕ to open the Formula Builder. From **Geometry**, open the **Measure** heading, select the PointToPoint() function, and click ✕ to open its property sheet.

b. Select the **Point 0** parameter, click ✕ to enter interactive reference mode, move to cell **C2**, hold the ○ button, and select the range **C2:D2** (the row and column centroid for the first blob, indexed 0).

c. Release ○, and then click ✕ to accept the reference, returning to the property sheet.

d. Select the **Point 1** parameter, click ✕, move to cell **C3**, hold ○, select **C3:D3** (the row and column centroid for the second blob), and click ✕ to accept.

e. Click **OK** to close the property sheet. PointToPoint() returns a Dist structure to store its results. It also emits the end points of a line segment, its length, and its angle from the image row axis:

A5=	PointToPoint(C2,D2,C3,D3,0)						
	A	B	C	D	E	F	G
0	Image						
1		Index	Row	Col	Angle	Color	Score	An
2	Blobs	0.000	212.423	94.055	175.136	1.000	0.000	71
3		1.000	212.376	464.398	176.920	1.000	0.000	66
4		Row0	Col0	Row1	Col1	Distance	Angle	
5	Dist	212.423	94.055	212.376	464.398	370.342	90.007	
6								
7								
8								
9		Distance (segment length)						

Blob 0, **Blob 1**, Distance (segment length) annotations appear beside the table.

The value in **F5** is measured distance between the two points. In this case, the input row and column values are in image coordinates. They are pixel offsets from the image origin. Therefore, the measured distance is the number of pixels between the two points. Converting the Blobs structure to World coordinates through TransBlobToWorld would transform the centroid row and column values to World coordinates. The results from PointToPoint(), reflecting its inputs, would also be in World units.

By default, In-Sight displays the graphics associated with the Dist structure only when you highlight it, as described in *Flyover Graphics*. To display the graphics all the time, set the function's **Show** parameter to **Show All**.

The procedures for using the other measurement functions are similar to those listed here.

COGNEX In-Sight

◄ ▲ ► Geometry: Function Reference

CircleFromPoints	Constructs a circle from three points. Returns a Circle structure. Emits the center row, center column, and radius.
CircleToCircle	Measures the shortest distance between two circles. Returns a Dist structure. Emits the segment's end points, distance, and angle. Distance is positive if the circles are separated, 0.0 if tangent or intersecting, negative if enclosed.
Dist structure	Stores a line segment defined by its end points, length, and an angle (from the image row axis, by default).
LineToCircle	Measures the shortest distance from a line to a circle. Returns a Dist structure. Emits the segment's end points, distance, and angle. If the line and circle intersect, emits the point(s) of intersection, and the distance is 0.00.
LineToLine	Measures the angle between two lines, in CCW degrees. Returns a Dist structure. Emits the intersection point, distance, and angle. Intersecting: distance is 0.00. Parallel: distance is positive, and the angle is 0, +180, or −180.
MidLineToMidLine	Measures the shortest distance between the midpoints of two line segments. Returns a Dist structure. Emits the segment's end points (the midpoints), distance, and angle.
PointToCircle	Measures the shortest distance from a point to a circle. Returns a Dist structure. Emits the segment's end points, distance, and angle. Distance is positive if the point is outside the circle, 0.0 if on it, negative if inside.
PointToLine	Measures the shortest distance from point to a line. Returns a Dist structure. Emits the segment's end points, distance, and angle.
PointToPoint	Measures the shortest distance between two points. Returns a Dist structure. Emits the segment's end points, distance, and angle.
PointToPointAngle	Returns the angle of a line segment with respect to the image row axis, in counterclockwise degrees.
PointToPointDistance	Returns the distance between two points.
SegmentFromLines	Constructs a line segment by averaging two line segments. Returns a Line structure. Emits the end points.

Data Access functions that get values from a Dist structure are GetAngle, GetCol, GetDistance, GetRow.

COGNEX In-Sight

◄ ▲ ► Geometry: CircleFromPoints function

Description	Constructs a circle from three points. Returns a Circle structure. Emits the center row, center column, and radius.
Heading	Geometry/Fit
Syntax	CircleFromPoints(Point 0, Point 1, Point 2, Show)

Inputs	Point 0	First point, row and column coordinates.
	—Row	
	—Column	
	Point 1	Second point, row and column coordinates.
	—Row	
	—Column	
	Point 2	Third point, row and column coordinates.
	—Row	
	—Column	
	Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	Circle	On success, a Circle structure.
Emits	Row	Center row, in the same coordinate system as the input points. Read by GetRow.
	Column	Center column, in the same coordinate system as the input points. Read by GetCol.
	Radius	The radius, in the same coordinate system as the input points. Read by GetRadius.
Errors		Invalid point, or the points are collinear and therefore do not define a circle.
Comments		None.
See Also		SegmentFromLines.

COGNEX In-Sight

◄ ▲ ► Geometry: CircleToCircle function

Description	Measures the shortest distance between two circles. Returns a Dist structure. Emits the segment's end points, distance, and angle. Distance is positive if the circles are separated, 0.0 if tangent or intersecting, negative if enclosed.
Heading	Geometry/Measure
Syntax	CircleToCircle(Circle 0, Circle 1, Show)
Inputs	Circle 0	First circle: center point and radius.
	—Row	
	—Column	
	—Radius	
	Circle 1	Second circle: center point and radius.
	—Row	
	—Column	
	—Radius	
	Show	Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Dist**	On success, a <u>Dist structure</u>.
Emits	**Row0**	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col0	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Row1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Distance	Length of the line segment, in the same coordinate system as the input points. Positive values indicate the circles are separated; 0.00 indicates tangent or intersecting circles; negative indicate that one circle encloses the other. Read by <u>GetDistance</u>.
	Angle	Angle of the line segment. Measured from the image row axis unless the input points are in World coordinates. Read by <u>GetAngle</u>.
Errors	Invalid parameter.	
Comments	Cases are:	

A   Points are end points. Distance is positive.

B   Points are single point of intersection and identical. Distance is 0.0.

C   Points are intersections. Distance is 0.0.

D   Points are end points. Distance is negative.

E   Points are intersection and identical. Distance is 0.0.

See Also   <u>LineToCircle</u>, <u>PointToCircle</u>, <u>PointToLine</u>

COGNEX In-Sight

◄ ▲ ► Geometry: Dist structure

	Value	Description	Access function
Description		Stores a line segment defined by its end points, length, and an angle (from the image row axis, by default).	
Values	Row 0	The first end point. In pixels from the image origin unless the input points are in World coordinates.	GetRow, index 0
	Col 0	The first end point. In pixels from the image origin unless the input points are in World coordinates.	GetCol, index 0
	Row 1	The second end point. In pixels from the image origin unless the input points are in World coordinates.	GetRow, index 1
	Col 1	The second end point. In pixels from the image origin unless the input points are in World coordinates.	GetCol, index 1
	Distance	The measured distance between the two end points. In pixels unless the input points are in World coordinates.	GetDistance
	Angle	Angle value, in counterclockwise degrees. Measured from the image row axis unless the input points are in World coordinates.	GetAngle
Comments	None.		
See Also	CircleToCircle, LineToCircle, LineToLine, MidLineToMidLine, PointToCircle, PointToLine, PointToPoint.		

COGNEX In-Sight

◄ ▲ ► Geometry: LineToCircle function

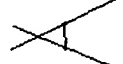|	
Description	Measures the shortest distance from a line to a circle. Returns a Dist structure. Emits the segment's end points, distance, and angle. If the line and circle intersect, emits the point(s) of intersection, and the distance is 0.00.
Heading	Geometry/Measure
Syntax	LineToCircle(Line, Circle, Show)
Inputs	**Line** A line. **Row 0** and **Column 0** are the first end point, and **Row 1** and **Column 1** are the second end point. In Pixel or World units.
	—Row0
	—Column0
	—Row1
	—Column1
	Circle A circle, defined by its center point and radius. In Pixel or World units.
	—Row
	—Column
	—Radius
	Show Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all

- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Dist**	On success, a <u>Dist structure</u>.
Emits	**Row0**	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col0	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Row1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Distance	Length of the line segment, in the same coordinate system as the input points. 0.00 indicates intersection. Read by <u>GetDistance</u>.
	Angle	Angle of the line segment. Measured from the image row axis unless the input points are in World coordinates. Read by <u>GetAngle</u>.
Errors		Invalid parameter.
Comments		LineToCircle() treats the segment defined by (Row0, Column0) and (Row1, Column1) as a line, projecting it if necessary to obtain its result points:
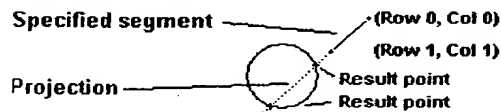
If the line and circle intersect, the returned distance is zero. The cases are:

A   Points are end points. Distance is positive.

B   Points are tangent and identical. Distance is 0.0.

C   Points are intersections. Distance is 0.0.

See Also	<u>CircleToCircle</u>, <u>LineToLine</u>, <u>MidLineToMidLine</u>, <u>PointToCircle</u>, <u>PointToLine</u>

COGNEX In-Sight

◄ ▲ ► Geometry: LineToLine function

| Description	Measures the angle between two lines, in counterclockwise degrees. Returns a Dist structure. Emits the intersection point, distance, and angle. Intersecting: distance is 0.00. Parallel: distance is positive, and the angle is 0, +180, or −180.
Heading	Geometry/Measure
Syntax	LineToLine(Line 0, Line 1, Show)
Inputs	**Line 0** A line. **Row 0** and **Column 0** are the first end point, and **Row 1** and
	—Row0 **Column 1** are the second end point. In Pixel or World units.
	—Column0
	—Row1
	—Column1
	Line 1 A line. **Row 0** and **Column 0** are the first end point, and **Row 1** and
	—Row0 **Column 1** are the second end point. In Pixel or World units.
	—Column0
	—Row1
	—Column1
	Show Selects the graphics to display, as described in *Flyover Graphics*: • Hide all • Result graphics only • Input and result graphics • Show all, which displays input, result, and chart (if any) graphics.
Stores	**Dist** On success, a Dist structure.
Emits	**NOTE** When the lines intersect, this function emits the same point twice, so Row0 = Row1 and Col0 = Col1.
	Row0 First end point or intersection, in the same coordinate system as the input points. Read by GetRow.
	Col0 First end point or intersection, in the same coordinate system as the input points. Read by GetCol.
	Row1 Second end point or intersection, in the same coordinate system as the input points. Read by GetRow.
	Col1 Second end point or intersection, in the same coordinate system as the input points. Read by GetCol.
	Distance Length of the line segment, in the same coordinate system as the input points: • Zero if the lines intersect. • A positive value if the lines are parallel, indicating their measured separation. Read by GetDistance.
	Angle Angle of the line segment: • The measured angle, −180 to +180 degrees, if the lines intersect. • One of 0.00, +180.00, or −180.00 if the lines are parallel. The direction of the lines determines the angle value returned:
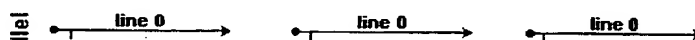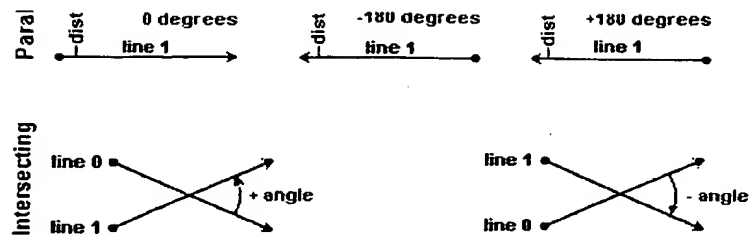
Measured from the image row axis unless the input points are in World coordinates. Read by GetAngle.

Errors   Invalid parameter.

Comments   LineToLine() treats both segments defined their (Row 0, Column 0) and (Row 1, Column 1) end points as lines, projecting them if necessary to obtain the intersection point. Cases are:

A   Points are intersection and identical. Distance is 0.0.

B   Angle is 0.0, indicating parallelism.
     Distance is positive, indicating separation.
     Points are endpoints of distance.

See Also   LineToCircle, MidLineToMidLine, PointToPointAngle

COGNEX In-Sight

◄ ▲ ► Geometry: MidLineToMidLine function

Description   Measures the shortest distance between the midpoints of two line segments. Returns a Dist structure. Emits the segment's end points (the midpoints), distance, and angle.

Heading   Geometry/Measure

Syntax   MidLineToMidLine(Line 0, Line 1, Show)

Inputs

Line 0   A line. **Row 0** and **Column 0** are the first end point, and **Row 1** and
—**Row0**   **Column 1** are the second end point. In Pixel or World units.
—**Column0**
—**Row1**
—**Column1**

Line 1   A line. **Row 0** and **Column 0** are the first end point, and **Row 1** and
—**Row0**   **Column 1** are the second end point. In Pixel or World units.
—**Column0**
—**Row1**
—**Column1**

**Show**   Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Dist**	On success, a <u>Dist structure</u>.
Emits	**Row0**	First midpoint, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col0	First midpoint, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Row1	Second midpoint, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col1	Second midpoint, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Distance	Length of the line segment, in the same coordinate system as the input points. Read by <u>GetDistance</u>.
	Angle	Angle of the line segment. Measured from the image row axis unless the input points are in World coordinates. Read by <u>GetAngle</u>.
Errors	Invalid parameter.	
Comments	Cases are:	

A   **Points are midpoints. Distance is positive.**

B   **Points are midpoints. Distance is positive.**

See Also   <u>LineToCircle</u>, <u>LineToLine</u>, <u>PointToLine</u>

COGNEX In Sight

◄ ▲ ► Geometry: PointToCircle function

Description	Measures the shortest distance from a point to a circle. Returns a Dist structure. Emits the segment's end points, distance, and angle. Distance is positive if the point is outside the circle, 0.0 if on it, negative if inside.
Heading	Geometry/Measure
Syntax	PointToCircle(Point, Circle, Show)
Inputs	**Point** A point. In Pixel or World units.
	—Row
	—Column
	Circle A circle, defined by its center point and radius. In Pixel or World units
	—Row
	—Column
	—Radius
	Show Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	Dist	On success, a <u>Dist structure</u>.
Emits	Row0	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col0	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Row1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Distance	Length of the line segment, in the same coordinate system as the input points. A positive value if the point is outside the circle, 0.0 if on the circle, negative if inside the circle. Read by <u>GetDistance</u>.
	Angle	Angle of the line segment. Measured from the image row axis unless the input points are in World coordinates. Read by <u>GetAngle</u>.
Errors	Invalid parameter.	
Comments	If the point and circle intersect, the measured distance is zero, and the end points are identical. Case are:	

A **Points are end points. Distance is positive.**

B **Points are identical. Distance is 0.0.**

C **Points are end points. Distance is negative.**

| See Also | <u>CircleToCircle</u>, <u>LineToCircle</u>, <u>PointToLine</u> |

COGNEX In-Sight

◄ ▲ ► Geometry: PointToLine function

Description	Measures the shortest distance from point to a line. Returns a Dist structure. Emits the segment's end points, distance, and angle.
Heading	Geometry/Measure
Syntax	PointToLine(Point, Line, Show)
Inputs	**Point** — A point. In Pixel or World units.
	—Row
	—Column
	Line — A line. **Row 0** and **Column 0** are the first end point, and **Row 1** and
	—Row0 — **Column 1** are the second end point. In Pixel or World units.
	—Column0
	—Row1
	—Column1
	Show — Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Dist**	On success, a <u>Dist structure</u>.
Emits	**Row0**	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col0	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Row1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Distance	Length of the line segment, in the same coordinate system as the input points. Zero if the point falls on the line. Read by <u>GetDistance</u>.
	Angle	Angle of the line segment. Measured from the image row axis unless the input points are in World coordinates. Zero if the point falls on the line. Read by <u>GetAngle</u>.
Errors		Invalid parameter.
Comments		PointToLine() treats the segments defined (Row 0, Column 0) and (Row 1, Column 1) as a line, projecting it if necessary to obtain the intersection point.

If the point falls on the line, the distance and angle are both zero. Cases are:

A **No projection. Result points are endpoints. Distance is positive.**

B **Segment projected. Result points are endpoints. Distance is positive.**

C **Point falls on the projected line. Distance is 0.0.**

See Also   <u>LineToCircle</u>, <u>LineToLine</u>, <u>MidLineToMidLine</u>, <u>PointToCircle</u>

COGNEX In-Sight

◄ ▲ ► Geometry: PointToPoint function

Description	Measures the shortest distance between two points. Returns a Dist structure. Emits the segment's end points, distance, and angle.
Heading	Geometry/Measure
Syntax	PointToPoint(Point 0, Point 1)

Inputs	Point 0 —Row —Column	A point. In Pixel or World units.
	Point 1 —Row —Column	A point. In Pixel or World units.
Returns	Dist	On success, a <u>Dist structure</u>.
Emits	Row0	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col0	First end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Row1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetRow</u>.
	Col1	Second end point or intersection, in the same coordinate system as the input points. Read by <u>GetCol</u>.
	Distance	Length of the line segment, in the same coordinate system as the input points. Zero if Point 0 = Point 1. Read by <u>GetDistance</u>.
	Angle	Angle of the line segment. Measured from the image row axis unless the input points are in World coordinates. Zero if Point 0 = Point 1. Read by <u>GetAngle</u>.
Errors	Invalid parameter.	
Comments	Differs from <u>PointToPointAngle</u> and <u>PointToPointDistance</u> by returning a Dist structure instead of a floating-point value.	
See Also	<u>LineToLine</u>, <u>MidLineToMidLine</u>, <u>PointToLine</u>, <u>PointToPointAngle</u>, <u>PointToPointDistance</u>, <u>*Getting Started with Geometric Measurement*</u>.	

COGNEX In-Sight

◄ ▲ ► Geometry: PointToPointAngle function

Description	Returns the angle of a line segment, in counterclockwise degrees, as a floating-point value.
Heading	Geometry/Measure
Syntax	PointToPointAngle(Point 0, Point 1)
Inputs	**Point 0** A point. In Pixel or World units. **—Row** **—Column** **Point 1** A point. In Pixel or World units. **—Row** **—Column**
Returns	Angle of the line segment, in counterclockwise degrees, as a floating-point value. Measured from the image row axis unless the input points are in World coordinates. Zero if Point 0 = Point 1.
Emits	Nothing.
Errors	Invalid parameter.
Comments	Differs from <u>PointToPoint</u> by returning a floating-point angle instead of a Dist structure.

See Also   LineToLine, MidLineToMidLine, PointToLine, PointToPoint.

COGNEX In-Sight

◄ ▲ ► Geometry: PointToPointDistance function

Description	Returns the distance between two points.
Heading	Geometry/Measure
Syntax	PointToPointDistance(Point 0, Point 1)
Inputs	**Point 0** A point. In Pixel or World units. **—Row** **—Column** **Point 1** A point. In Pixel or World units. **—Row** **—Column**
Returns	Length of the line segment, in the same coordinate system as the input points. A floating-point value. Zero if Point 0 = Point 1.
Emits	Nothing.
Errors	Invalid parameter.
Comments	Differs from PointToPoint by returning a floating-point angle instead of a Dist structure.
See Also	CircleToCircle, LineToCircle, LineToLine, MidLineToMidLine, PointToCircle, PointToLine, PointToPoint, PointToPointAngle.

COGNEX In-Sight

◄ ▲ ► Geometry: SegmentFromLines function

Description	Constructs a line segment by averaging two line segments. Returns a Line structure. Emits the end points.
Heading	Geometry/Fit
Syntax	SegmentFromLines(Line 0, Line 1, Show)
Inputs	**Line 0** A line. **Row0** and **Column0** are the first end point, and **Row1** and **—Row0** **Column1** are the second end point. In Pixel or World units. **—Column0** **—Row1** **—Column1** **Line 1** A line. **Row0** and **Column0** are the first end point, and **Row1** and **—Row0** **Column1** are the second end point. In Pixel or World units. **—Column0** **—Row1** **—Column1** **Show** Selects the graphics to display, as described in *Flyover Graphics*:

- Hide all
- Result graphics only
- Input and result graphics
- Show all, which displays input, result, and chart (if any) graphics.

Stores	**Line**	On success, a Line structure.
Emits	**Row0**	First end point, row or X coordinate, in the same coordinate system as the input points. Read by GetRow.
	Col0	First end point, column or Y coordinate, in the same coordinate system as the input points. Read by GetCol
	Row1	Second end point, row or X coordinate, in the same coordinate system as the input points. Read by GetRow.
	Col1	Second end point, column or Y coordinate, in the same coordinate system as the input points. Read by GetCol.
Errors	Invalid parameter.	
Comments	None.	
See Also	CircleFromPoints.	

COGNEX In-Sight

◄ ▲ ► Graphics: Overview

The Graphics functions draw graphics on the image or in worksheet cells. They are grouped three headings, Controls, Displays, and Image. The controls—buttons, list boxes, check boxes and so on—are graphical inputs stored in worksheet cells, commonly as part of an operator interface. Controls remain editable even when the worksheet is locked, so that the operator can control the application. The displays are output graphics drawn in a worksheet cell, to monitor a value for application development or deployment. Displays include charts that plot the history of a value (to determine correct parameter settings, or for process monitoring) and a status indicator (color coded for Pass/Fail/Warning). The Image functions draw overlay graphics—arcs, circles, crosses, and so on—on the image.

This section introduces the Graphics functions, discussing the following topics:

- Getting Started with Image Graphics
- Getting Started with Worksheet Graphics
- Getting Started with Charts
- Graphics Function Reference.

The graphics are drawn on the image by a nondestructive overlay. They influence the displayed image but not the underlying pixel data. Attributes available for all image graphics are a name (a label), a color, and a toggle to enable or disable the graphic.

COGNEX In-Sight

◄ ▲ ► Graphics: Using Image Graphics

In this topic, you draw a circle over an image, yielding this result:

A2	PlotCircle(A1,B1,C1, "My Circle",1,1)				
	A	B	C	D	E
0	@Image				
1	100.000	300.000	25.000		
2	@Plot				
3					
4					

— Circle
— Name

The steps for drawing other shapes such as arcs, points, and lines are similar to those described here.

► **To Draw a Circle**

1. You define a circle with three values, its center row, center column, and radius. For this exercise, you place static values in three cells and subsequently refer to them to define the circle:
   a. From cell **A1**, click and hold ✕, select **Value** from the Enter menu, edit the value 100.0 into

the cell, and click ✕ to accept it. This value will be the center row coordinate.
  b. From cell **B1**, repeat the process to put 300.0 into **B1**, to serve as the center column coordinate.
  c. From cell **C1**, repeat the process to put 25.0 into **C1**. This value will be the radius, in pixels.

Any of these values can be static, computed by formulas, or stored in a Circle structure.

2. From cell **A2**, click ✕ to open the Formula Builder.
3. From the **Graphics** category, select **Image**, and then open the PlotCircle property sheet.
4. Connect PlotCircle() to the row, column, and radius values by creating relative references:
  a. Select the **Circle** heading, and then click and hold ✕, opening the Enter menu.
  b. Select **Relative** and release ✕, placing In-Sight in interactive reference mode.
  c. Move the marquee to cell **A1**, hold down the ○ button, and click cursor Right to select the range **A1:C1**. Release ○, and then ✕ to select the references, returning to the property sheet.

You could, alternatively, create absolute references.

5. From the **Name** heading, click ✕, opening the Text Entry dialog.
6. Edit the string "My Circle", then click **OK** to accept the name, returning to the property sheet.
7. From the **Color** heading, select **Dark Red**, and click ✕ to accept it, returning to the property sheet.
8. Click **OK**, accepting the parameters and closing the property sheet. In-Sight draws the specified circle:

A2=	PlotCircle(A1,B1,C1,"My Circle",1,1)				
	A	B	C	D	E
0	⊕Image				
1	100.000	300.000	25.000		
2	⊕Plot				
3					
4					

— Circle
— Name

9. With cell **A2** highlighted, In-Sight draws the circle as a flyover graphic. It therefore has the default color, not the specified color (Dark Red). To view the circle exactly as defined, move away from **A2**.

COGNEX In·Sight

◄ ▲ ► Graphics: Using Worksheet Graphics

In this topic, you put a button in the worksheet that acquires a new image when clicked:

A1=	Button("Click To Acquire")			
	A	B	C	D
0	⊕Image			
1	⊡Click To Acquire			

Appendix I - Page **191**

The steps for defining other worksheet graphics are similar to those described here.

▶ **To Put a Button in a Cell**

1. From cell **A1**, click ✕ to open the Formula Builder.
2. From the **Graphics** category, select **Worksheet**, and then open the <u>Button</u> property sheet.
3. On the **String** parameter, click ✕, opening the <u>Text Entry</u> dialog.
4. Edit the string "Click to Acquire".
5. Click **OK** to accept the name, closing the property sheet and placing a ▭ graphic in **A1**.
    NOTE In-Sight displays as much of the button label as it can without hiding an occupied cell. If the
          label is too long to fit in one cell and the adjacent cells are unoccupied, In-Sight displays the
          entire label. If the adjacent cells are occupied, it truncates the label at the first occupied cell.
6. Move to cell **A0**, which contains an Image structure that stores the acquired image.
7. Click ✕ to open the <u>AcquireImage</u> property sheet.
8. On the **??Internal Trigger??** parameter, click ✕ to switch to interactive reference mode.
9. Move the <u>marquee</u> to **A1**, and then click ✕ to accept the reference. Click **OK** to close the property
    sheet.
10. Move to **A1**.
11. Click ✕ on the ▭ graphic. In-Sight acquires a new image.
12. Internally, cell **A1**'s value changes from 0.00 to 1.00 when clicked. In-Sight passes the change to
    AcquireImage(), which responds by acquiring a new image. To view the change in value:
    a. Move to cell **A2**.
    b. Click and hold ✕, select **Relative** from the Enter menu, and release ✕.
    c. Move the marquee to **A1**, the Button() cell, and click ✕ to accept the reference. **B1** displays
        0.00, the current (unclicked) value of Button().
    d. Move to **A1** and click the button. **B1** displays 1.00, the value when clicked.

COGNEX In-Sight

◀ ▲ ▶ Graphics: Using Charts

In this topic, you create a *chart*, a graphic that records a changing value over time. The result looks like this:

	A	B	C	D	E	F	G	H	I	J
A3=	Chart(A0,G2,10,"My Chart",0,0)									
0	Image									
1		Thresh	NDark	NBright	AveDark	AveBright	Contrast			
2	Hist.	86.000	3769.000	6231.000	41.717	126.988	85.270			
3	85.50	My Chart	Name			Charted value				
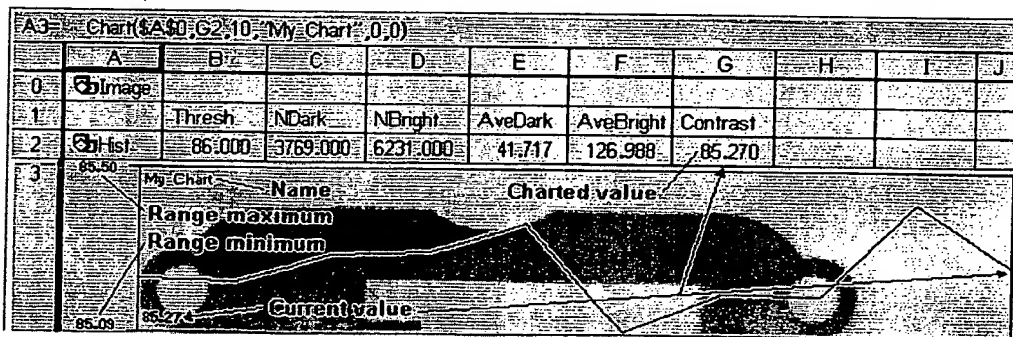
During setup, you can use charts to help determine correct parameter settings. In the deployed application, you can use charts to improve process monitoring.

▶ **To Create a Chart**

1. Acquire an appropriate image. For this exercise:
    a. Open the part sample image, print it, and return to this topic. Place the printed image under the camera.
    b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
    c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

    To acquire a new image, hold ☐ and click ✕.

2. Before you chart a value, you have to have a value to chart. A convenient value for this exercise is the contrast result from a histogram. To obtain it:
    a. From cell **A2**, click ✕ to open the Formula Builder.
    b. From **Vision Processing**, open the Histogram heading, and then open the ExtractHistogram property sheet.
    c. Click ✕ on the Region heading, switching to the interactive Region cursor. Use the Control Pad to adjust the region so that it includes a feature in the image.
    d. Click ✕ to accept the region, and then click **OK** to close the property sheet. In-Sight computes the histogram and emits the contrast and other values.
3. From cell **A3**, click ✕ to open the Formula Builder. From **Graphics**, open the **Worksheet** heading, and then open the Chart property sheet.
4. Select the **Value** parameter and click ✕ switching to interactive reference mode.
5. Move the marquee to cell **G2**, the contrast value. Click ✕ to accept the reference and return to the property sheet.
6. Select the **Chart Name** parameter and click ✕ In the resulting Text Entry dialog, edit the name "My Chart" and click ✕ to accept.
7. Leave the **Range Min** and **Range Max** values unchanged. When both are 0.00 (the default), Chart() automatically determines the range based on recent values. A large increase or decrease in the charted value might change the automatic ranges. Setting either **Range Min** or **Range Max** to a non-0.00 value overrides the automatic range in favor of the requested range.
8. Click **OK** to close the property sheet. In-Sight replaces the worksheet row containing Chart() with a chart graphic. At this point, the chart contains only one value, the current histogram contrast.
    NOTE If the row containing the chart already contains a value, then the chart graphic hides the value. The chart does not destroy any data, but you can't see other values in the row because the chart is in the way. You can move from cell to cell in the hidden row, keeping track of the column through the highlight in the column header. References to the hidden cell remain intact.

    This situation creates a risk of confusion. For example, if you clear the Chart() cell and accidentally select one or more hidden cells that contain values, then you lose the data in the hidden cells. Consider adopting the habit of creating charts in empty rows.
9. Click and release △ to open the System menu. Select **Triggers** to open the Triggers menu, and then click **Manual**. In-Sight acquires a new image and updates the chart, which now contains two values.

Small differences in lighting usually cause the histogram contrast value to change slightly from image to image. The chart reflects these differences.

10. Click **Manual** at least ten more times to fill the chart with values. (You can set the number of time increments to plot in the Chart property sheet. The default, unchanged here, is 10.) After the chart fills, the oldest value rolls off the left end.

11. From the Triggers menu, enable **Continuous**. In-Sight acquires images as quickly as possible, updating the chart to reflect them.

COGNEX In·Sight

◄ ▲ ► Graphics: Function Reference

Button	Puts a labeled button control, ▭, in a cell. Returns 1.00 when clicked, 0.00 otherwise. Remains adjustable when the worksheet is locked.
Chart	Puts a chart display in a cell, to plot the history of a value.
CheckBox	Puts a labeled check box control in a cell. When ON (☒), returns 1.00; when OFF (☐), 0.00. Typically referenced to set a toggled value. Remains adjustable when the worksheet is locked.
ColorLabel	Puts a text string of a specified color in a cell of a specified color.
Dialog	Creates a view window opened by a labeled button.
EditFloat	Puts a numeric edit box control in a cell. Returns a floating-point value constrained to a specified range. Remains adjustable when the worksheet is locked.
EditInt	Puts a numeric edit box control in a cell. Returns an integer constrained to a specified range. Remains adjustable when the worksheet is locked.
EditString	Puts a text edit box control in a cell. Returns a string limited to a specified maximum number of characters. Remains adjustable when the worksheet is locked.
ListBox	Puts a list box control in a cell. Returns the index of the selected list item. Remains adjustable when the worksheet is locked.
Plot structure	Stores a graphic.
PlotArc	Draws an arc graphic from a starting point to an ending point around a specified center point.
PlotCircle	Draws a circle graphic with a specified center, radius, name, and color.
PlotCross	Draws a cross mark graphic with a specified center, angle, height, width, name, and color.
PlotLine	Draws a line graphic between two points, with a specified name and color.
PlotPoint	Draws a point graphic at a specified point in the image.
PlotRegion	Draws a rectangular region graphic, potentially rotated and curved.
PlotString	Draws a string graphic at a specified point.
Status	Puts a status display in a cell. Green if positive; yellow if 0.00; red if negative.

COGNEX In·Sight

◄ ▲ ► Graphics: Button function

Description	Puts a labeled button control, ▢, in a cell. Returns 1.00 when clicked, 0.00 otherwise.	
Heading	Graphics/Controls	
Syntax	Button(Name)	
Inputs	**Name**	A text label for the button.
	NOTE	In-Sight displays as much of the button label as it can without hiding an occupied cell. If the label is too long to fit in one cell and the adjacent cells are unoccupied, In-Sight displays the entire label. If the adjacent cells are occupied, it truncates the label at the first occupied cell.
Returns	1.00 when clicked; 0.00 otherwise.	
Emits	Nothing.	
Errors	Invalid parameter.	
Comments	To graphically control the value of a function parameter—for example as part of an operator interface—reference a Button() cell from that parameter. The control remains adjustable even when the worksheet is locked.	

The referencing parameter typically is a "flag" toggle, trigger, or other parameter that accepts 0.00 for OFF or 1.00 for ON, making the button a graphical proxy for the flag. Button() returns a 0.00 or a 1.00 even if the referencing item is not a flag. In that case, the result of clicking the button is the same as setting the item to 0.00 or 1.00 by any other method.

Button() is similar to CheckBox() but nonpersistent. You commonly use Button() to trigger one-time operations such as training a model.

NOTE After you define a button, you cannot click X on its cell as a shortcut for opening the Button() property sheet. To reopen the property sheet, you must click O to open the Edit menu, and then select Formula.

See Also	Chart, CheckBox, ColorLabel, Dialog, EditInt, ListBox, Status, *Getting Started with Worksheet Graphics*.

COGNEX In·Sight

◄ ▲ ► Graphics: Chart function

Description	Puts a chart display in a cell, to plot the history of a value.	
Heading	Graphics/Displays	
Syntax	Chart(Event, Value, Number, Name, Range Min, Range Max)	
Inputs	**Event**	An event that forces an update of the chart. Must be a reference to a cell containing AcquireImage (the default), Button, or Event.
	Value	The value to plot, typically a cell reference.
	Number	Number of time increments to plot.
	Name	Name of the chart, a string.
	Range	Range to display. If Min and Max both equal 0.0, Chart() automatically
	—Min	assigns a range, based on recent values. If either Max or Min is non-0.0,
	—Max	then Chart() observes the requested values.

Returns	0.00 on success.
Emits	Nothing.
Errors	Invalid parameter.

Example

	A	B	C	D	E	F	G	H	I	J
0	ⓘImage									
1		Thresh	NDark	NBright	AveDark	AveBright	Contrast			
2	ⓘHist	86.000	3769.000	6231.000	41.717	126.988	85.270			
3										

A3 = Chart(A0, G2, 10, "My Chart", 0, 0)
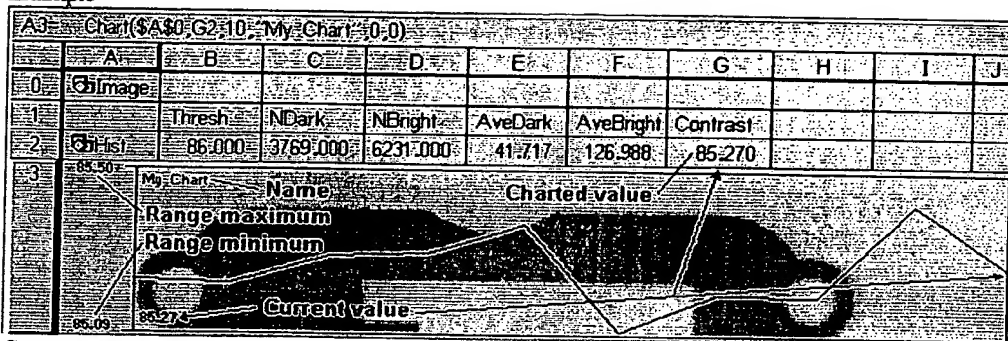
Comments   If the row containing the chart already contains a value, then the chart graphic hides the value. The chart does not destroy any data, but you can't see other values in the row because the chart is in the way. You can move from cell to cell in the hidden row, keeping track of the column through the highlight in the column header. References to the hidden cell remain intact.

This situation creates a risk of confusion. For example, if you clear the Chart() cell and accidentally select a hidden cell, then you lose the data in the hidden cell. Consider adopting the habit of creating charts in empty rows.

See Also   ColorLabel, Dialog, Status, *Getting Started with Charts.*

COGNEX In·Sight

◀ ▲ ▶ Graphics: CheckBox function

Description	Puts a labeled check box control in a cell. When ON (☒), returns 1.00; when OFF (☐), 0.00. Typically referenced to set a toggled value.
Heading	Graphics/Controls
Syntax	Checkbox(Name)
Inputs	**Name** A text label for the check box.
Returns	1.00 when ON, 0.00 when OFF.
Emits	Nothing.
Errors	Invalid parameter.
Comments	To graphically control the value of a parameter—for example as part of an operator interface—reference a CheckBox() cell from that parameter. The control remains adjustable even when the worksheet is locked.

The referencing parameter is typically a toggle that accepts 0.00 for OFF or 1.00 for ON, making the check box a graphical proxy for the toggle. CheckBox() emits a 0.00 or a 1.00 even if the referencing item is not a toggle. In that case, the result of clicking the

Appendix I - Page **196**

check box is the same as setting the item to 0.00 or 1.00 by any other method.

Button() is similar to CheckBox() but non-persistent.

NOTE   After you define a check box, you cannot click ✕ on its cell as a shortcut for
       opening the CheckBox() property sheet. To reopen the property sheet, you must
       click ○ to open the Edit menu, and then select Formula.

See Also   Button, Dialog, EditInt, ListBox, *Getting Started with Worksheet Graphics*.

COGNEX In-Sight

◄ ▲ ► Graphics: ColorLabel function

Description	Puts a text string of a specified color in a cell of a specified color.
Heading	Graphics/Displays
Syntax	ColorLabel(Name, ForeColor, BackColor)
Inputs	**Name** A string to display.
	ForeColor A color for the string.
	BackColor A color for the cell.
Returns	0.00 on success.
Emits	Nothing.
Example	

Red Text	Blue	Green

Errors	Invalid parameter.
Comments	None.
See Also	Button, Chart, CheckBox, EditInt, ListBox, *Getting Started with Worksheet Graphics*.

COGNEX In-Sight

◄ ▲ ► Graphics: Dialog function

Description	Creates a a customized worksheet view accessed by a labeled button.
Heading	Graphics/Controls
Syntax	Dialog(Name, Position, Force Defaults, Formula Bar, Column Headers, Row Headers, Grid Lines, Scroll Range, Top-Left Cell, Active Cell)
Inputs	**Name** A text label for the button that switches to the worksheet view.
	Position On-screen location and size of the worksheet view. **Row** and **Col**
	—Row define the top-left point by offsets from the Image origin, in pixels. **High**
	—Column and **Wide** are the height and width, in pixels.
	—High
	—Wide
	Force Forces the worksheet view to look and act like the full-size standard
	Defaults worksheet. When ON, Force Defaults overrides subsequent parameters.

Appendix I - Page **197**

Formula Bar	ON enables the formula bar.
Column Headers	ON enables the column headers (A through Z).
Row Headers	ON enables the row headers (0 through 999).
Grid Lines	ON enables the grid lines drawn on the worksheet.
Scroll Range	Restricts scrolling in the view window to a specified range of cells.
—Enable	**Enable** ON limits scrolling, and OFF permits scrolling for the entire
—Start	worksheet. When ON, **Start** is the top-left cell accessible by scrolling
—End	and **End** is the bottom-right cell accessible by scrolling.
Top-Left Cell	The worksheet cell placed in the top-left corner of the view window.
Active Cell	The worksheet cell highlighted after switching to the view window.

Returns	????
Emits	Nothing.
Errors	Invalid parameter.
Comments	Dialog() lets you create one or more window-like views into the worksheet, typically as part of an operator interface. By placing other controls such as buttons or list boxes inside the viewed region, you can create dialog-box-like areas for operator use. For each view:

1. Define the on-screen location and size of the window through the row and column of the top-left point and a height and width in pixels.
2. Enable or disable various worksheet attributes (formula bar, row headers, column headers, gridlines, and so on.)
3. Define the range of worksheet cells visible through the window, the top-left visible cell, the active cell, and so on.

After defining the view window, Dialog() puts a labeled button in the cell. Clicking the button opens the view window:
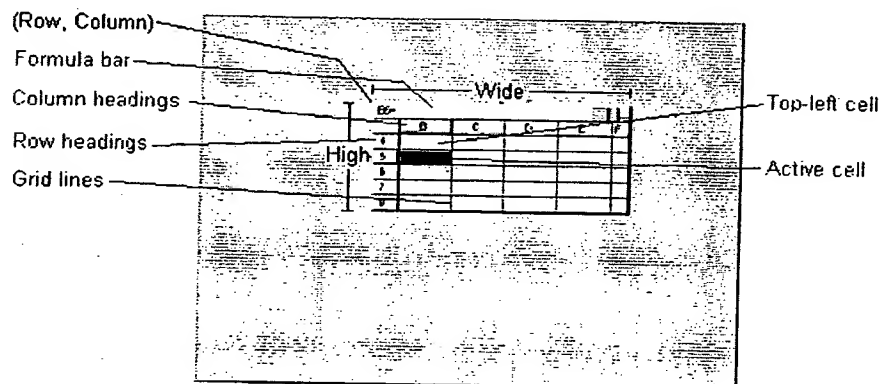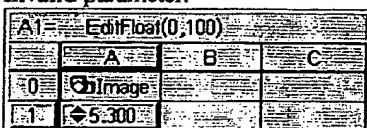
NOTE	After you define a view window, you cannot click ✕ on its cell (the activation button) as a shortcut for opening the Dialog() property sheet. To reopen the property sheet, you must click ○ to open the Edit menu, and then select Formula.
See Also	Button, Chart, Status, *Getting Started with Worksheet Graphics*.

COGNEX In Sight

◄ ▲ ► **Graphics: EditFloat function**

Description	Puts a numeric edit box control in a cell. Returns a floating-point value constrained to a specified range. Remains adjustable when the worksheet is locked.
Heading	Graphics/Controls
Syntax	EditFloat(Min, Max)
Inputs	**Min** The minimum value for the numeric selection graphic.
	Max The maximum value for the numeric selection graphic.
Returns	A floating-point value between Min and Max.
Emits	Nothing.
Errors	Invalid parameter.
Example	

A1=	EditFloat(0,100)		
	A	**B**	**C**
0	Image		
1	5.300		

Comments	This function gives you a way to let the operator input a floating-point value even when you lock the worksheet. The selection graphic remains adjustable even when the worksheet is locked.

> **NOTE**   After you define a numeric selection graphic, you cannot click ✕ on its cell as a shortcut for opening its property sheet. To reopen the property sheet, you must click ○ to open the Edit menu, and then select Formula.

See Also	Button, Chart, CheckBox, ColorLabel, Dialog, EditInt, EditString, ListBox, Status, *Getting Started with Worksheet Graphics*.

COGNEX In Sight

◄ ▲ ► **Graphics: EditInt function**

Description	Puts a numeric edit box control in a cell. Returns an integer constrained to a specified range. The value remains editable even when the worksheet is locked.
Heading	Graphics/Controls
Syntax	EditInt(Min, Max)
Inputs	**Min** The minimum value for the integer selection graphic.
	Max The maximum value for the integer selection graphic.
Returns	An integer between Min and Max.
Emits	Nothing.
Errors	Invalid parameter.

Example

A1=	EditInt(0,10)		
	A	B	C
0	🖼Image	📷Event	
1	◆ 3		

Comments   This function gives you a way to let the operator input an integer even when you lock the worksheet. Remains adjustable even when the worksheet is locked.

NOTE   After you define a integer selection graphic, you cannot click ✕ on its cell as a shortcut for opening its property sheet. To reopen the property sheet, you must click O to open the Edit menu, and then select Formula.

See Also   Button, Chart, CheckBox, ColorLabel, Dialog, EditFloat, EditString, ListBox, Status, _Getting Started with Worksheet Graphics_.

═══ **COGNEX In·Sight**

◄ ▲ ► Graphics: EditString function

Description   Puts a text edit box control in a cell. Returns a string constrained to a specified range. The value remains editable even when the worksheet is locked.

Heading   Graphics/Controls

Syntax   EditString(MaxStringLen)

Inputs   **MaxStringLen**   Maximum length for the user-edited string, in characters.

Returns   A string up to MaxStringLen characters long.

Emits   Nothing.

Errors   Invalid parameter.

Example

A1=	EditString(10)		
	A	B	C
0	🖼Image		
1	◆My String		

Comments   This function gives you a way to let the operator input a text string even when you lock the worksheet. Remains adjustable even when the worksheet is locked.

NOTE   After you define a string selection graphic, you cannot click ✕ on its cell as a shortcut for opening its property sheet. To reopen the property sheet, you must click O to open the Edit menu, and then select Formula.

See Also   Button, Chart, Dialog, EditFloat, EditInt, CheckBox, ColorLabel, ListBox, Status, _Getting Started with Worksheet Graphics_.

═══ **COGNEX In·Sight**

◄ ▲ ► Graphics: ListBox function

Description	Puts a list box control in a cell. Returns the index of the selected list item.
Heading	Graphics/Controls
Syntax	ListBox(String1, ..., StringN)
Inputs	**String1 ...** Labels for the list items. Two or more strings or references to cells
	StringN containing strings.
Returns	Index of the selected list item. The first item is 0.00.
Emits	Nothing
Errors	Invalid parameter.
Example	

A2= ListBox(A1,B1,C1)		
A	**B**	**C**
0 ❸Image		
1 first	second	third
2 ▶ first		
3 second		
4 third		

Comments To graphically control a function parameter that accepts a list of values—for example as part of an operator interface—reference a cell containing ListBox() from that parameter. The control remains adjustable even when the worksheet is locked.

From ListBox(), assign names for the enumerated parameter values (**String1**, and so on). ListBox() then creates a list box associated with the cell. When the user selects the list box, it pops up, so that the user can select one of the enumerated items.

NOTE   After you define a list box, you cannot click ✕ on its cell as a shortcut for opening the ListBox() property sheet. To reopen the property sheet, you must click ◯ to open the Edit menu, and then select Formula.

See Also   Button, Chart, CheckBox, ColorLabel, Dialog, EditInt, Status, *Getting Started with Worksheet Graphics*.

COGNEX In·Sight

◄ ▲ ► Graphics: Plot structure

Description	Stores a graphic.
Values	Plot has no internal values that are read individually by Data Access functions.
Comments	Returned by graphics structures that store a graphic.
See Also	PlotArc, PlotCircle, PlotCross, PlotLine, PlotPoint, PlotRegion, PlotString, *Getting Started Plotting Graphics*.

COGNEX In·Sight

◄ ▲ ► Graphics: PlotArc function

Description	Draws an arc graphic from a starting point counterclockwise to an ending point around a specified center point. If necessary, fits an arc by averaging the center-to-end point distances.
Heading	Graphics/Image
Syntax	PlotArc(Arc, Name, Color, Show)
Inputs	**Arc** The arc, defined by row and column coordinates for the center point,
	—**Center Row** starting end point, and ending end point, in pixels from the image origin.
	—**Center Col**
	—**Start Row**
	—**Start Col**
	—**End Row**
	—**End Col**
	Name A text label for the graphic. A string or a reference to one.
	Color A color, selected from a list.
	Show ON to enable the arc graphic; OFF to disable it.
Returns	**Plot** On success, a <u>Plot structure</u>, which stores the graphic.
Emits	Nothing.
Errors	Invalid parameter.
Comments	• If the center point and end points define a circle, then PlotArc() draws the arc exactly as defined. If the end points do not fall on a circle with the specified center, then PlotArc() averages the two center-to-end point distances and fits an arc from the resulting circle.
	• PlotArc() draws the arc counterclockwise from the Start point to the End point:

Yes　　　　　　No

Counterclockwise　Clockwise
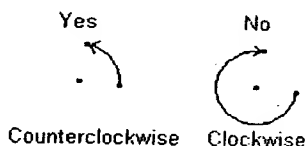
See Also	<u>PlotCircle</u>, <u>PlotCross</u>, <u>PlotLine</u>, <u>PlotPoint</u>, <u>PlotRegion</u>, <u>PlotString</u>, _Getting Started Plotting Graphics_.

COGNEX In·Sight

◄ ▲ ► Graphics: PlotCircle function

Description	Draws a circle with a specified center, radius, label, and color.
Heading	Graphics/Image
Syntax	PlotCircle(Circle, Name, Color, Show)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(72) Inventor: MCGARRY, John; 12395 SW Corylus, Portland, OR 97224 (US).

(74) Agent: POWSNER, David, J.; Choate, Hall & Stewart, Exchange Place, 53 State Street, Boston, MA 02109 (US).

[Continued on next page]

(54) Title: MACHINE VISION ANALYSIS UTILIZING A SPREADSHEET INTERFACE
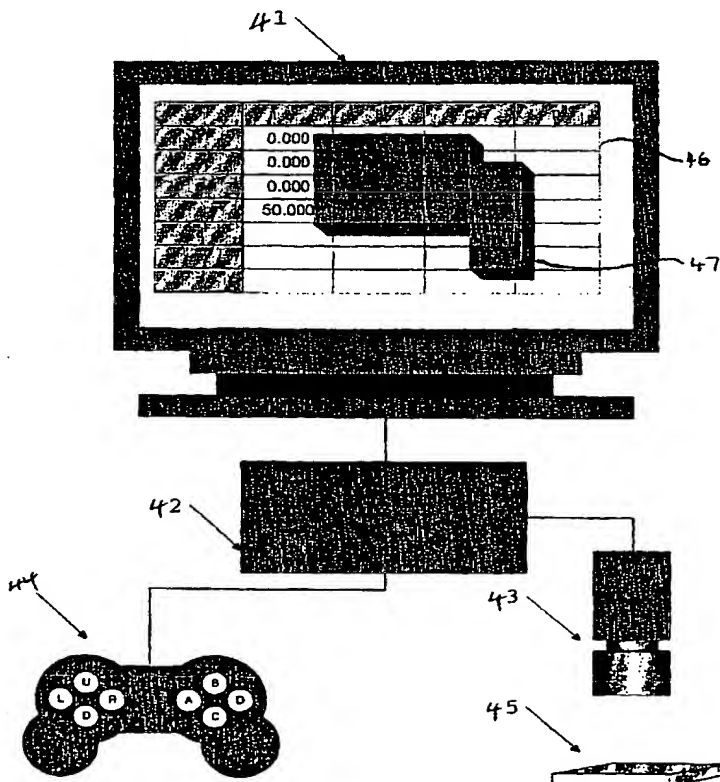
(57) Abstract: A user interface is provided for programming machine vision applications that includes an image window for displaying an image to be analyzed (41), and a spreadsheet (46) for analyzing the image displayed with the image window. A hand-held control pad (44) is used to navigate over the spreadsheet (46), thereby selecting cells of the spreadsheet. The control pad is additionally used for menu item selection, and for spreadsheet input. The spreadsheet is semi-transparent, and is adapted to perform conditional cell execution, as well as operate on time interval data. When the control pad indicates a specific cell of the spreadsheet, a change occurs in the image to be displayed (47). Buffered outputs are utilized for circular references or recursive operations. Complex vision objects are also instantiated as a single cell in the spreadsheet.

WO 01/11445 A3

INTERNATIONAL SEARCH REPORT

	International application No. PCT/US00·22383

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :Please See Extra Sheet.
US CL :707/503; 345/505; 712/34

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/503; 345/505; 712/34

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

ACM search terms: machine vision, spreadsheet, real time analysis

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,410,649 A (GOVE) 25 April 1995, whole document.	1-40
A	US 5,926,822 A (GARMAN) 20 July 1999, whole document.	1-40
A	US 5,574,930 A (HALVERSON, JR et al) 12 November 1996, whole document.	1-40

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 JANUARY 2001	26 FEB 2001

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	HEATHER HERNDON
Facsimile No. (703) 305-3230	Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)★

A. CLASSIFICATION OF SUBJECT MATTER:
IPC (7):

G06F 15/00, 15/76, 15/80, 17/00, 17/21, 17/24

Comments   To get the current online state, use the Event() function. ??How??
See Also   None.

COGNEX In Sight

◄ ▲ ► Text: Overview

The Text functions manipulate text strings in various ways. For example, you can find text inside a string, replace text inside a string with new text, convert a string to lowercase or UPPERCASE, compare two strings, and so on. Each string can contain up to to 255 alphanumeric characters (bytes) and must be enclosed in quotes. The Text functions are in general case-sensitive. Common uses include manipulating strings for serial communications or for on-screen display.

This section introduces the Text functions, discussing the following topics:

- Using the Text Functions
- Text Functions Reference.

Any cell in the In-Sight worksheet cell can hold a string, which you insert or edit though the Formula Builder, as described in *Entering Text*. A reference to an empty cell as a string is generally interpreted as a null string. For functions that process more than one string, the maximum number of characters per cell is 2048.

Each character in a string has a zero-based index. The first character is number 0.00, the second is number 1.00, and so on. Text operations that depend on the location of a character inside a string express the locations in terms of these zero-based indices.

NOTE   Functions that return strings cannot act as direct arguments for certain classes of functions. This restriction affects functions that return structures (such as ExtractBlobs(), FindPatterns(), and many others), the lookup functions If() and Choose(), and the worksheet graphics functions such as Button() and ListBox(). In these situations, place the string function in a distinct cell and refer to that cell from the complex function.

COGNEX In-Sight

◄ ▲ ► Text: Using the Text Functions

In this topic, you concatenate and format a string with the Stringf, a representative Text function. The result looks like this:

A3=	Stringf("%s","Contrast is ",G2)					
	A	B	C	D	E	F	G
0	⊞Image						
1		Thresh	NDark	NBright	AveDark	AveBright	Contrast
2	⊞Hist	79.000	5969.000	4031.000	37.479	117.610	80.130
3	Contrast is 80.13078						

Other Text functions require similar steps.

▶ **To Format a String with Stringf()**

1. To illustrate Stringf() in a machine vision context, start by setting up worksheet that computes a value. A convenient value for this exercise is the contrast result from a histogram. To obtain it:
    a. Acquire an appropriate image. During development, you typically place a part sample under the test stand and snap a picture of it. Alternatively, to adjust the camera height, aperture, and focus, use live mode.
    b. From cell A2, click ✕ to open the Formula Builder.
    c. From **Vision Processing**, open the Histogram heading, and then open the ExtractHistogram property sheet.
    d. Click ✕ on the Region heading, switching to the interactive Region cursor. Use the Control Pad to adjust the region to include a feature in the image.
    e. Click ✕ to accept the region, and then click **OK** to close the property sheet. In-Sight computes the histogram and emits the Contrast and other values.
2. From cell A3, click ✕ to open the Formula Builder. From the **Text** heading, click Stringf().
3. To specify the first parameter (the format), click the 🔲 button to open the Text Entry dialog.
4. Click ○ to toggle between UPPERCASE and lowercase. From the uppercase character set, select %, the percent sign. Click ○ to switch to lowercase, click s, and then click **OK** to accept the %s format.
5. Click the comma (,) character, which is the delimiter between the first and second parameters.
6. To specify the second parameter, again click 🔲 and similarly edit the string "Contrast is ". Note the trailing space. When done, click **OK**.
7. Click comma to place the delimiter between the second and third parameters.
8. For the third parameter, click 🔲, move the marquee to cell G2 (the Contrast value), and click ✕ to accept the reference.
9. Enter a closing parenthesis character to mark the end of the formula.
10. Click **OK** to accept the finished formula. Stringf() concatenates its parameters and formats the result string in the %s format:

A3=	Stringf("%s","Contrast is ",G2)					
	A	B	C	D	E	F	G
0	Image						
1		Thresh	NDark	NBright	AveDark	AveBright	Contrast
2	Hist	79.000	5969.000	4031.000	37.479	117.610	80.130
3	Contrast is 80.13078						

The %s format displays the value at a higher precision than that of the emitted formula in G2.

COGNEX In-Sight

◀ ▲ ▶ **Text: Function Reference**

<u>Concatenate</u>	Concatenates a variable number of text or numeric parameters. Returns the concatenated string.
<u>Exact</u>	Compares two strings and returns True (1.00) if they are identical. Case sensitive.
<u>Find</u>	Finds a string inside another string, starting at a specified index. Returns the zero-based index of the first matching character. Case sensitive.
<u>Left</u>	Returns the left-most characters from a text string.
<u>Len</u>	Returns the number of characters in a text string.
<u>Lower</u>	Converts a text string to lowercase; returns the lowercase string.
<u>Mid</u>	Returns a specified number of characters from a specified location in a text string.
<u>Replace</u>	Replaces characters inside a text string with new characters. Returns the edited string.
<u>Right</u>	Returns the right-most characters from a text string.
<u>Strcspn</u>	Searches a string for the first character included in a specified set of characters. Returns the zero-based index of the first matching character. Case sensitive.
<u>Stringf</u>	Formats a string in one of these formats: %c, %d, %e, %E, %f, %o, %s, %u, %x, or %X.
<u>Strspn</u>	Searches a string for the first character not in a specified set of characters. Returns the zero-based index of the first non-matching character. Case sensitive.
<u>Strtol</u>	Converts a string that starts with an integer to its integer value, including hexadecimal and octal values. Ignores subsequent non-integer characters. Truncates floating-point inputs.
<u>Substitute</u>	Replaces text in a string with new text. Returns the edited string. Replaces all instances unless you specify an instance number.
<u>Token</u>	Returns a specified instance of text from a list, with a specified delimiter.
<u>Trim</u>	Removes leading and trailing spaces from a text string. Also removes extra internal spaces, leaving one space between words. Returns the edited string.
<u>Upper</u>	Converts a text string to UPPERCASE; returns the uppercase string.
<u>Value</u>	Converts a text string that starts with with an integer to its integer value. Ignores subsequent non-integer characters.

COGNEX In Sight

◄ ▲ ► Text: Concatenate function

Description	Concatenates its parameters, returning the concatenated string.
Heading	Text
Syntax	Concatenate(Arg1, [ArgN, ...])
Inputs	**Arg1 ...** One or more text strings or numeric values. Any numeric values are automatically converted to strings.
Returns	The formatted string, concatenated from **Arg1 ... ArgN**.
Emits	Nothing.
Errors	• Invalid parameter. • Result string exceeds 255 characters.
Examples	• Concatenate("Combine ", "these", "strings") returns "Combine these strings". • Concatenate("Converts ", "numerics ", 1, 2, 3) returns "Converts numerics 123".

Comments	None.
See Also	Strtol, Value

COGNEX In·Sight

◄ ▲ ► Text: Exact function

Description	Compares two text strings and returns True (1.00) if they are identical. Case sensitive.
Heading	Text
Syntax	Exact(Text1, Text2)
Inputs	**Text1** A text string or a reference to one.
	Text2 A text string or a reference to one.
Returns	**1.00** The strings are exactly identical.
	0.00 The strings differ.
Emits	Nothing.
Errors	Invalid parameter. For example, Text1 or Text2 is not a string.
Example	• Exact("Same", "Same") returns 1.00.
	• Exact("Same", "Different") returns 0.00.
	• Exact("DIFFERENT", "different") returns 0.00.

Comments	None.
See Also	Len

COGNEX In·Sight

◄ ▲ ► Text: Find function

Description	Finds a string inside another string, starting at a specified index. Returns the zero-based index of the first matching character. Case sensitive.
Heading	Text
Syntax	Find(FindText, SearchText, [StartChar])
Inputs	**FindText** The text to find. A text string or a reference to one.
	SearchText The string to search for FindText. A text string or a reference to one.
	StartChar The starting location in SearchText. A zero-based character index.
Returns	• Index of the first matching character. Zero-based.
	• −1, indicating "not found."
Emits	Nothing.
Errors	• Invalid string parameter.
	• StartChar < 0.

Example	• Find("d", "abcdefg") returns 3.00.
	• Find("D", "abcdefg") returns −1.00 (not found).
	• Find("c", "abcabcabc", 4) returns 5.00.

| Comments | The −1 return value lets you distinguish between "not found" and "invalid parameter." |
| See Also | Replace, Strcspn, Strspn, Substitute, Token |

COGNEX In-Sight

◄ ▲ ► Text: Left function

Description	Returns the left-most characters from a text string.
Heading	Text
Syntax	Left(Text1, NumChars)
Inputs	**Text1** A text string or a reference to one.
	NumChars The number of characters to return.
Returns	The left-most characters. A string.
Emits	Nothing.
Errors	• Invalid string parameter.
	• NumChars < 0.

Example	Left("This is a string", 4) returns "This".
Comments	If **NumChars** exceeds the number of characters actually in **Text1**, Left() returns the entire string, not an error.
See Also	Mid, Right

COGNEX In-Sight

◄ ▲ ► Text: Len function

Description	Returns the number of characters in a text string.
Heading	Text
Syntax	Len(Text1)
Inputs	**Text1** A text string or a reference to one.
Returns	Length of Text1, in characters.
Emits	Nothing.
Errors	Invalid string parameter.
Example	Len("This is a string") returns 16.00.

| Comments | None. |
| See Also | Exact |

=== **COGNEX** In-Sight

◄ ▲ ► Text: Lower function

Description	Converts a text string to lowercase; returns the lowercase string.
Heading	Text
Syntax	Lower(Text1)
Inputs	**Text1** A text string or a reference to one.
Returns	The lowercase text string.
Emits	Nothing.
Errors	Invalid string parameter.
Example	Lower("AbCdEfG") returns "abcdefg".
Comments	None.
See Also	Upper.

=== **COGNEX** In-Sight

◄ ▲ ► Text: Mid function

Description	Returns a specified number of characters from a specified location in a text string.
Heading	Text
Syntax	Mid(Text1, StartChar, NumChars)
Inputs	**Text1** The source string. A text string or a reference to one.
	StartChar The starting location in **Text1**. A zero-based character index.
	NumChars The number of characters to extract.
Returns	The characters extracted from **Text1**.
Emits	Nothing.
Errors	• Invalid string parameter.
	• StartChar or NumChars < 0.
Example	• Mid("This is a string", 0, 7) returns "This is".
	• Mid("This is a string", 1, 8) returns "his is a".
Comments	None.
See Also	Left, Right

=== **COGNEX** In-Sight

◄ ▲ ► Text: Replace function

Description	Replaces text inside a string with new text. Returns the edited string.
Heading	Text
Syntax	Replace(SrcText, StartChar, NumChars, NewText)
Inputs	**SrcText** Source text. A text string or a reference to one.
	StartChar Index of the first character to replace. Zero-based.
	NumChars Number of characters in to replace.
	NewText Text to insert in **Text1**. A text string or a reference to one.
Returns	The edited string. **Text1** modified by **Text2**.
Emits	Nothing.
Errors	• Invalid string parameter.
	• StartChar or NumChars < 0.
	• Result string exceeds 255 characters.
Example	• Replace ("This is a string", 0, 4, "That") returns "That is a string."
	• Replace ("This is a string", 8, 1, "a changed") returns "This is a changed string."
Comments	None.
See Also	Find, Strcspn, Strspn, Substitute, Token

COGNEX In-Sight

◄ ▲ ► Text: Right function

Description	Returns the right-most characters from a text string.
Heading	Text
Syntax	Right(Text1, NumChars)
Inputs	**Text1** A text string or a reference to one.
	NumChars The number of characters to return.
Returns	The right-most characters. A string.
Emits	Nothing.
Errors	• Invalid string parameter.
	• NumChars < 0.
Example	Right("This is a String", 4) returns "ring".
Comments	If **NumChars** exceeds the number of characters actually in **Text1**, Right() returns the entire string, not an error.
See Also	Left, Mid

COGNEX In-Sight

◄ ▲ ► Text: Strcspn function

Description	Searches a string for the first character included in a specified set of characters. Returns the zero-based index of the first matching character. Case sensitive.
Heading	Text
Syntax	Strcspn(SrcText, CharList)
Inputs	**SrcText** A string, or a reference to one, to check against **CharList**.
	CharList A set of characters to test for inclusion. Order and duplication do not matter. No delimiter. For example, "ABC" means "A" or "B" or "C".
Returns	• Index of the first character in **SrcText** that is present in **CharList**. Zero based.
	• If no characters in **SrcText** are present in **CharList**, returns the length of SrcText.
Emits	Nothing.
Errors	Invalid string parameter.
Example	• Strcspn("This is a string", "QWERTY") returns 0.00.
	• Strcspn("This is a string", "qwerty") returns 11.00.
	• Strcspn("abc", "xyz") returns 3.0, which is the length of the first parameter.
Comments	Similar to strcspn() in ANSI C. Strspn(), in contrast, searches a string for the first character **excluded** from a specified set of characters.
See Also	Find, Replace, Strspn, Substitute, Token

COGNEX In-Sight

◄ ▲ ► Text: Stringf function

Description	Formats a string in one of these formats: %c, %d, %e, %E, %f, %o, %s, %u, %x, or %X.
Heading	Text
Syntax	Stringf([%format,] Text1)
Inputs	**%format** A text string representing a format specifier, one of:

%c	Char.
%d	Decimal integer.
%e or %E	Scientific notation.
%f	Floating-point. Default (except that trailing zeros are trimmed).
%o	Octal.
%s	String.
%u	Unsigned integers.
%x or %X	Hexadecimal.

Text1 ...	One or more text strings. Numeric values are automatically converted to strings.
Returns	The formatted string. **Text1** in **%format**.
Emits	Nothing.

Errors	• Invalid parameter.
	• Result string exceeds 255 characters.
Examples	• ??CHECK THESE EXAMPLES
	• Stringf("%d",3) returns "3".
	• Stringf("decimal:%d is hex:%X",255,255) returns "decimal:255 is hex:FF".
	• Stringf("PI = %f",Pi) returns "PI = 3.141593".
	• Stringf("Char code of '%c' is '%d'","A","A") returns "Char code of 'A' is '65'".
Comments	• In-Sight's Stringf() function is similar to the C-language sprintf() function. The %format specifiers therefore support several options controlling precision and other attributes. Their more general syntax is:

$$\%[\textit{flag}][\textit{min-field}[\textit{.precision}]\textit{conversion-letter}$$

where:

- o *flag* is a minus sign (–) to left-justify or a zero (0) to pad. C's # flag is unsupported.
- o *min-field* is a decimal integer that defines a minimum width in characters.
- o *.precision* is a decimal integer that controls precision: the minimum number of digits for %d, %i, %o, %u, and %x; the digits to the right of the decimal for %e, %E, and %f; and the maximum number of characters for %s.
- o *conversion letter* is one of the conversion operations (c, d, e, E, f, o, s, u, x, or X).

C's "size" option is unsupported. For more information about formatting, consult a C-language text such as *C A Reference Manual* by Harbison and Steele.
- • To convert from text to numbers, use Strtol() or Value().

See Also	Concatenate, Strtol, Value

COGNEX In·Sight

◄ ▲ ► Text: Strspn function

Description	Searches a string for the first character not in a specified set of characters. Returns the zero-based index of the first non-matching character. Case sensitive.
Heading	Text
Syntax	Strspn(SrcText, CharList)
Inputs	**SrcText** The string, or a reference to one, to check against **CharList**.
	CharList A set of characters to test for exclusion. Order and duplication do not matter.

Returns
- Index of the first character in **SrcText** that is absent from **CharList**. Zero-based.
- If all characters in in **SrcText** are also in **CharList**, Strspn() returns the length of **SrcText**.

Emits   Nothing.
Errors   Invalid parameter.
Example
- Strspn("This is a string", "xyz") returns 0.00.
- Strspn("This is a string", "QWERTY") returns 1.00.
- Strspn("This is a string", "This") returns 4.00.
- Strspn("This is a string", "This is a string") returns 16.00 (the length of the first parameter).
- Strspn("This is", "This is a string") returns 7.00 (the length of the first parameter).

Comments   Similar to strspn() in ANSI C. Strcspn(), in contrast, searches a string for the first character **included** in specified set of characters.
See Also   Find, Replace, Strcspn, Substitute, Token

COGNEX In-Sight

◄ ▲ ► Text: Strtol function

Description   Converts a string that starts with an integer to its integer value, including hexadecimal and octal values. Ignores subsequent non-integer characters. Truncates floating-point inputs.
Heading   Text
Syntax   Strtol(IntText)
Inputs   **IntText**   A text string that represents a numeric value, or a reference to one. Strings starting with 0x or 0X are interpreted as hexadecimal, and strings starting with zero (0) are interpreted as octal.
Returns   The integer value obtained from **IntText**, displayed as a floating-point value.
Emits   Nothing.
Errors   Invalid string parameter. For example, IntText does not start with a digit.
Example
- Strtol("1234") returns the value 1234, displayed as 1234.000.
- Strtol("1234abcd") returns the value 1234, displayed as 1234.000.
- Strtol("1234.567") returns the value 1234, displayed as 1234.000.

Comments
- In-Sight's Strtol() function is similar to the C-language strtol() function. It consequently interprets a string starting with 0x or 0X as a hexadecimal value and a string starting with zero (0) as an octal value. Its handling of a leading zero differs from that of the worksheet. In the worksheet, In-Sight interprets "010" as the decimal value ten. But Strtol() interprets 010 as the octal value 10, equal to the decimal value eight. In-Sight retains this aspect of the C-language strtol() function for the benefit of those needing to read octal values and familar with the C-language convention.
- Value() is similar to Strtol(), except that Value() does not recognize the 0, 0x, and 0X prefixes as octal and hexadecimal.

- To convert from numbers to text, use Stringf().

See Also   Concatenate, Stringf, Value

COGNEX In:Sight

◄ ▲ ► Text: Substitute function

Description	Replaces text in a string with new text, returning the edited string. Replaces all instances unless you specify an instance number.
Heading	Text
Syntax	Substitute(SrcText, OldText, NewText, [Instance])
Inputs	**SrcText** The source string. A text string or a reference to one.
	OldText The sequence of characters to be replaced inside **SrcText**. A text string or a reference to one.
	NewText A string of characters to substitute for **OldText**.
	Instance An instance number. A zero-based index: 0 for the first match, 1 for the second, and so on. If the parameter is omitted, Substitute() replaces all instances.
Returns	The edited string.
Emits	Nothing.
Errors	• Invalid parameter.
	• Result string exceeds 255 characters.
Example	• Substitute ("abcdefabcdef", "abc", "XYZ") returns "XYZdefXYZdef".
	• Substitute ("abcdefabcdef", "abc", "XYZ", 1) returns "abcdefXYZdef".
	• Substitute ("abcdefabcdef", "abc", "XYZ", 5) returns "abcdefabcdef". (No change, because the index 5 is not present in the source string.)
Comments	None.
See Also	Find, Replace, Strcspn, Strspn, Token

COGNEX In:Sight

◄ ▲ ► Text: Token function

Description	Returns a specified instance of text from a list, with a specified delimiter.
Heading	Text
Syntax	Token(Text1, Delimiter, Instance)
Inputs	**Text1** A string containing a list of text items, separated by **Delimiter**.
	Delimiter A string defining a delimiter within **Text1**.
	Instance The instance number to select. Zero-based: 0 for the first instance, 1 for the second, and so on.

Returns	The selected instance. A string.
Emits	Nothing.
Errors	• Invalid string parameter.
	• Instance < 0.
Example	• ??Token("a b, c, d", " ,", 2) returns "c".
	• ??Token(" a b, c, d", " ,", 2) returns "c".
Comments	None.
See Also	Find, Replace, Strcspn, Strspn, Substitute

COGNEX In-Sight

◄ ▲ ► Text: Trim function

Description	Removes leading and trailing spaces from a text string. Also removes extra internal spaces, leaving one space between words. Returns the edited string.
Heading	Text
Syntax	Trim(Text1)
Inputs	**Text1** A text string or a reference to one.
Returns	The edited string. **Text1** without extra spaces.
Emits	Nothing.
Errors	Invalid string parameter.
Example	• Trim(" Remove outside spaces ") returns "Remove outside spaces".
	• Trim("Remove inside spaces") returns "Remove inside spaces".
Comments	None.
See Also	Find, Replace, Substitute, Token

COGNEX In-Sight

◄ ▲ ► Text: Upper function

Description	Converts a text string to UPPERCASE; returns the uppercase string.
Heading	Text
Syntax	Upper(Text1)
Inputs	**Text1** A text string or a reference to one.
Returns	The UPPERCASE text string.
Emits	Nothing.
Errors	Invalid string parameter.
Example	Upper("aBcDeFg") returns "ABCDEFG".
Comments	None.
See Also	Lower.

COGNEX In-Sight

◄ ▲ ► Text: Value function

Description	Converts a text string that starts with an integer to its integer value. Ignores subsequent non-integer characters.
Heading	Text
Syntax	Value(NumText)
Inputs	**IntText** A text string that represents a number, or a reference to one.
Returns	The numeric value obtained from **IntText**. A floating-point quantity.
Emits	Nothing.
Errors	Invalid string parameter. For example, **IntText** does not start with a digit.
Example	- Value("1234") returns the value 1234, displayed as 1234.000 - Value("1234abcd") returns the value 1234, displayed as 1234.000 - Value("1234.567") returns 0.00 ??Said changed. Check this one.??
Comments	Similar to Strtol(), but does not convert octal or hexadecimal. To convert from numbers to text, use Stringf().
See Also	Concatenate, Stringf, Strtol

COGNEX In-Sight

◄ ▲ ► Data Access: Overview

The Vision Data Access functions return individual values from the structures generated by other In-Sight functions. They underly the automatically emitted formulas. You can usually obtain result values from the emitted formulas and ignore the Data Access functions, which are listed in the Formula Builder under **Advanced**. In-Sight offers them to provide access to values that aren't emitted (for example, the maximum and minimum row and column values for a blob) and to give you more control over the extracted data.

This section introduces the Data Access functions, discussing the following topics:

- Getting Started with Data Access
- Data Access Function Reference.

The exact result returned by a particular data access function depends on the referenced structure. For example, in general GetMax() returns the maximum value from a structure—so, when pointed at a Delay structure, it returns the largest value currently in the buffer.

Many structures store feature data extracted from images. Each item has a zero-based index and a set of measured values. Most Data Access functions have an index parameter to specify the particular item of interest. The indexes default to 0.00, so omitting the index yields the first item in the structure. In a few situations, you supply two indices. For example, each edge in an Edges structure has two row values, one for each end point. When GetRow() refers to an Edges structure, it therefore requires two indexes: one to specify a particular edge, and another to specify either the first or second end point.

COGNEX In-Sight

◄ ▲ ► Data Access: Getting Started

In this topic, you use the GetRow function to get a row coordinate—a value already available as an emitted formula—from a Blobs structure. You then use the GetNFound function to get the number of blobs—a value not emitted and otherwise unavailable. The result looks like this:

Six blobs found

Two not found

GetRow(A2,5)
(same as emitted
formula)

GetNFound(A2)
(returns number
actually found)

The techniques shown here apply to all Data Access functions.

▶ **To Get Data Values from a Structure**

1. Acquire an appropriate image. For this exercise:
   a. Open the <u>part sample image</u>, print it, and return to this topic. Place the printed image under the camera.
   b. Click and release △, select **Live** from the System menu, and click ✕ to enter *live mode*. Interactively adjust the image location, camera height, aperture, and focus until satisfied with the image.
   c. Click △ twice to exit live mode, acquire the last image, and return to the worksheet.

   To acquire a new image at any time, hold ☐ and click ✕.

2. Extract blob data from the image:
   a. From cell **A2**, click ✕ to open the <u>Formula Builder</u>. From **Vision Processing**, select **Blob**, and then open the <u>ExtractBlobs</u> property sheet.
   b. To define the region of interest, click the **Region** heading. In-Sight hides the worksheet and displays the <u>Region cursor</u>. Using the Control Pad, surround the part image, which has at most six blobs (four small round holes, one larger rectangular hole, and the entire perimeter). Click ✕ to return to the property sheet.
   c. Increase the **Number To Find** value to "8" (more than the number expected).
   d. Click **OK** to extract the blobs. ExtractBlobs() analyses the image for blobs that fit the selection criteria, draws an outline around them, and emits a block of formulas containing individual blob data values:

   Blobs object (stores feature data)
   Emitted formulas
   #ERR (no data)
   Region graphic

   Each Blob has an index, starting with 0 (displayed as "0.00"), used to specify a particular blob when you call a Data Access function. Here, ExtractBlobs() sought eight blobs (the Number to Find value) but actually found only six that met the selection criteria. Therefore, blobs 0 through 5 contain measured values, but blobs 6 and 7 are empty, as indicated by #ERR.

3. Show the equivalence of the emitted formulas and the Data Access functions:
   a. On cell **A10**, click ✕ to open the Formula Builder. Open **Advanced**, move to **Vision Data Access**, open the **Blobs** heading, select <u>GetRow</u>, and click ✕. GetRow() does not have a property sheet. Instead, you edit its two parameters directly from the Formula Builder.
   b. The first parameter is a reference to structure to serve as the data source. To create a <u>relative</u>

reference, click [⊞], move the marquee to A2 (the Blobs structure), and click ✕ to accept the reference.

c. Enter a comma (,) character as the delimiter between the first and second parameters.

d. The second parameter identifies the blob of interest through its index. The first blob is blob 0, so the sixth is blob 5. To select it, enter the digit 5.

e. Enter a close-parenthesis character to mark the end of the formula, and then click **OK** to accept it. GetRow() returns the row value for the sixth blob to **A10**. The value should equal the value in **C7**, which contains an emitted formula for the sixth blob's row. The Data Access functions underlie the emitted formulas, which are automatically added to the worksheet as a convenient way to expose the data in structures. There is no difference between the emitted formulas and equivalent formulas built by hand with the Data Access functions.

4. Some structures contain values not automatically emitted to the worksheet. For example, the Blobs structure stores the number of blobs actually found. To get this value:

a. From **A11**, open the Formula Builder. If necessary, open **Advanced**, move to **Vision Data Access**, open the **Blobs** heading, select GetNFound, and click ✕ Like GetRow(), GetNFound() does not have a property sheet.

b. To create a reference to the Blobs structure, click [⊞], move the marquee to **A2**, and click ✕ Because GetNFound() has only one parameter, In-Sight supplies the close-parenthesis character.

c. Click **OK**. Here, GetNFound() returns 6.00, indicating that ExtractBlobs() actually found six blobs that met the selection criterion:

Six blobs found
Two not found
GetRow(A2,5) (same as emitted formula)
GetNFound(A2) (returns number actually found)

COGNEX In Sight

◄ ▲ ► Data Access: Function Reference

GetAngle	Returns an angle value, in counterclockwise degrees, from a specified structure.
GetArea	Returns an area value, in pixels, from a specified structure.
GetCol	Returns a column coordinate from a structure, in Pixel or World units, reflecting the structure.
GetColor	Returns the color value, 0.0 for black or 1.0 for white, for a specified blob.
GetCount	Returns a count value from a specified structure.
GetCurve	Returns a curvature value, in counterclockwise degrees, from a structure.
GetDistance	Returns a distance value (the length of a line segment) from a structure.
GetElongation	Returns an elongation value (the ratio of major to minor axes) for a specified blob.
GetHigh	Returns a height value, in pixels, from a structure.
GetHoles	Returns the number of holes (background-colored areas) in a specified blob.
GetMax	Returns the maximum value from a specified structure.
GetMaxCol	Returns the maximum (right-most) column coordinate from a structure.
GetMaxRow	Returns the maximum (bottom-most) row coordinate from a structure.
GetMean	Returns the average (arithmetical mean) from a specified structure.
GetMin	Returns the minimum value from a specified structure.
GetMinCol	Returns the minimum (left-most) column coordinate from a specified structure.
GetMinRow	Returns the minimum (top-most) row coordinate from a specified structure.
GetNFound	Returns the number-found value from a specified structure.
GetPerimeter	Returns the perimeter, in pixels, from a specified structure.
GetPixelValue	From an Image structure, returns the grayscale value for a specified point.
GetRadius	Returns the radius of a specified circle or arc, in pixels.
GetRow	Returns a row coordinate from a structure, in Pixel or World units, reflecting the structure.
GetScale	Returns a scale value (the size of a matched instance, as a percentage of the model) from a specified structure.
GetScore	Returns a score value, 0 through 100, from a specified structure.
GetSDev	Returns the standard deviation value from a specified structure.
GetSpread	Returns the spread value (a rough measure of compactness) for a specified blob.
GetSum	Returns a sum value from a specified structure.
GetValue	Returns the number of values from a specified structure.
GetWide	Returns a width value, in pixels, from a structure.

COGNEX In-Sight

◄ ▲ ► Data Access: GetAngle function

Description	Returns an angle value, in counterclockwise degrees, from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetAngle(Struct, [Index])
Inputs	**Struct** A reference to a structure that contains an angle value.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	Angle, in counterclockwise degrees. Measured from the image row axis or the World X axis, reflecting the values in **Struct**.
Emits	Nothing.

Errors	?Invalid structure or index?.
Comments	Available from the Blobs, Cross, Dist, Fixture, Patterns, and Region structures. See each structure's documentation for details about the returned values.
See Also	GetCol, GetCurve, GetHigh, GetRow, GetWide.

COGNEX In-Sight

◄ ▲ ► Data Access: GetArea function

Description	Returns an area value, in pixels, from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetArea(Struct, [Index])
Inputs	**Struct** A reference to a structure containing an area value.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	Area. ?Always in pixels, even if the source structure is in World units.??.
Emits	Nothing.
Errors	Invalid structure or index.
Comments	Available from the Blobs structure. See its documentation for details about the returned value.
See Also	Blobs structure, GetColor, GetElongation, GetHoles, GetPerimeter, GetSpread.

COGNEX In-Sight

◄ ▲ ► Data Access: GetCol function

Description	Returns a column coordinate from a structure, in Pixel or World units, reflecting the structure.
Heading	Advanced/Vision Data Access
Syntax	GetCol(Struct, [Index1], [Index2])
Inputs	**Struct** A reference to a structure containing a column value.
	Index1 An zero-based index into **Struct**, identifying the item of interest.
	Index2 Edges only. A zero-based index identifying the end point: 0 for the first end point or 1 for the second end point.
Returns	Column value. In pixels from the image origin or from the World X axis, reflecting the source structure.
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the Blobs, Circle, Cross, Dist, Edges, Fixture, Line, Point, Patterns, and Region structure. See each structure's documentation for details about the returned value.
See Also	GetAngle, GetCurve, GetHigh, GetRow, GetWide.

COGNEX In-Sight

◄ ▲ ► Data Access: GetColor function

Description	Returns the color value, 0.0 for black or 1.0 for white, for a specified blob.
Heading	Advanced/Vision Data Access
Syntax	GetColor(Blobs, [Index])
Inputs	**Blobs** A reference to a Blobs structure. Each blob has a color value.
	Index A zero-based index into the Blobs structure. Default = 0, the first item.
Returns	**0.0** for black
	1.0 for white.
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the Blobs structure. For details about the returned value, see its documentation.
See Also	Blobs structure, GetArea, GetElongation, GetHoles, GetPerimeter, GetSpread.

COGNEX In-Sight

◄ ▲ ► Data Access: GetCount function

Description	Returns a count value from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetCount(Struct)
Inputs	**Struct** A reference to a Delay structure.
Returns	Delay structure: the number of values actually in the buffer.
Emits	Nothing.
Errors	Invalid structure or range.
Comments	For details about the returned value, see the documentation for the Delay structure.
See Also	Delay structure

COGNEX In-Sight

◄ ▲ ► Data Access: GetCurve function

Description	Returns a curvature value, in counterclockwise degrees, from a structure.
Heading	Advanced/Vision Data Access
Syntax	GetCurve(Struct)
Inputs	**Struct** A reference to a structure containing a curve value.
Returns	Angle of curvature, in counterclockwise degrees.
Emits	Nothing.
Errors	??Invalid structure.
Comments	Available from the Region structure. See its documentation for details about the returned value.
See Also	GetAngle, GetCol, GetHigh, GetRow, GetWide.

COGNEX In-Sight

◄ ▲ ► Data Access: GetDistance function

Description	Returns a distance value (the length of a line segment) from a structure.
Heading	Advanced/Vision Data Access
Syntax	GetDistance(Struct)
Inputs	**Struct** A reference to a Dist stucture.
Returns	Distance. In Pixel or World units, reflecting the source structure.
Emits	Nothing.
Errors	??Invalid structure.
Comments	Available from the Dist structure. See its documentation for details about the returned value.
See Also	Dist structure.

COGNEX In-Sight

◄ ▲ ► Data Access: GetElongation function

Description	Returns an elongation value (the ratio of major to minor axes) for a specified blob.
Heading	Advanced/Vision Data Access
Syntax	GetElongation(Blobs, [Index])
Inputs	**Blobs** A reference to a Blobs structure.
	Index A zero-based index into the Blobs structure. Default = 0, the first blob.
Returns	Elongation, the ratio of the major axis to the minor axis.
Emits	None.
Errors	Invalid structure or index.
Comments	For details about the elongation value, see the documentation for the Blobs structure.
See Also	Blobs structure, GetArea, GetColor, GetHoles, GetPerimeter, GetSpread.

COGNEX In-Sight

◄ ▲ ► Data Access: GetHigh function

Description	Returns a height value, in pixels, from a structure.
Heading	Advanced/Vision Data Access
Syntax	GetHigh(Struct)
Inputs	**Struct** A reference to a structure containing a height value.
Returns	Height. Always in pixels.
Emits	Nothing.
Errors	Invalid structure.
Comments	Available from the Cross and Region structures. See each structure's documentation for details about the returned values.
See Also	GetAngle, GetCol, GetCurve, GetRow, GetWide.

COGNEX In-Sight

◄ ▲ ► Data Access: GetHoles function

Description	Returns the number of holes (background-colored areas) in a specified blob.
Heading	Advanced/Vision Data Access
Syntax	GetHoles(Blobs, [Index])
Inputs	**Blobs** A reference to a Blobs structure.
	Index A zero-based index into the Blobs structure. Default = 0, the first blob.
Returns	The number of holes, an integer.
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the Blobs structure. See its documentation for details about the returned value.
See Also	Blobs structure, GetArea, GetColor, GetElongation, GetPerimeter, GetSpread.

COGNEX In-Sight

◄ ▲ ► Data Access: GetMax function

Description	Returns the maximum value from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetMax(Struct)
Inputs	**Struct** A reference to a <u>Delay</u> structure.
Returns	Delay structure: the largest value currently in the buffer.
Emits	Nothing.
Errors	Invalid structure or range.
Comments	Available from the <u>Delay</u> structure, which describes the returned value.
See Also	<u>DelayLine</u>, <u>GetMin</u>.

COGNEX In-Sight

◄ ▲ ► Data Access: GetMaxCol function

Description	Returns the maximum (right-most) column coordinate from a structure.
Heading	Advanced/Vision Data Access
Syntax	GetMaxCol(Struct, [Index])
Inputs	**Struct** A reference to a structure containing an maximum column value.
	Index A zero-based index into Struct. Default = 0, the first item.
Returns	The maximum column coordinate. Always in Pixel coordinates (because the maximum column value in World coordinates might be a completely different point on the blob perimeter).
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the <u>Blobs structure</u>. See its documentation for details about the returned value.
See Also	<u>GetCol</u>, <u>GetMaxRow</u>, <u>GetMinCol</u>, <u>GetMinRow</u>, <u>GetRow</u>.

COGNEX In-Sight

◄ ▲ ► Data Access: GetMaxRow function

Description	Returns the maximum (bottom-most) row coordinate from a structure.
Heading	Advanced/Vision Data Access
Syntax	GetMaxRow(Struct, [Index])
Inputs	**Struct** A reference to a structure containing a maximum row value.
	Index A zero-based index into Struct. Default = 0, the first item.
Returns	The bottom-most row coordinate. Always in Pixel coordinates (because the maximum row value in World coordinates might be a completely different point on the blob perimeter).
Emits	Nothing.
Errors	???Invalid structure or index.

Comments	Available from the <u>Blobs structure</u>. See its documentation for details about the returned value.
See Also	<u>GetCol</u>, <u>GetMaxCol</u>, <u>GetMinCol</u>, <u>GetMinRow</u>, <u>GetRow</u>.

═══════════════════════════ COGNEX In-Sight

◄ ▲ ► Data Access: GetMean function

Description	Returns the average (arithmetical mean) from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetMean(Struct, [??Range Start, Range End])
Inputs	**Struct** A <u>Delay structure</u>, which stores the buffer generated by DelayLine().
Returns	The mean value of the buffer.
Emits	Nothing.
Errors	??Invalid structure.
Comments	Available from the <u>Delay</u> structures, which describes the returned values.
See Also	<u>Delay</u>, <u>DelayLine</u>

═══════════════════════════ COGNEX In-Sight

◄ ▲ ► Data Access: GetMin function

Description	Returns the minimum value from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetMin(Struct, [Range Start, Range End])
Inputs	**Struct** A reference to a <u>Delay</u> structure.
Returns	Delay structure: the smallest value currently in the buffer.
Emits	Nothing.
Errors	Invalid structure or range.
Comments	For details about the returned value, see the documentation for the <u>Delay</u> structure.
See Also	<u>DelayLine</u>, <u>GetMax</u>.

═══════════════════════════ COGNEX In-Sight

◄ ▲ ► Data Access: GetMinCol function

Description	Returns the minimum (left-most) column coordinate from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetMinCol(Struct, [Index])
Inputs	**Struct** A reference to a structure containing an minimum column value.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	The left-most column coordinate. Always in Pixel coordinates (because the minimum column value in World coordinates might be a completely different point on the blob perimeter).
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the Blobs structure. See its documentation for details about the returned value.
See Also	GetCol, GetMaxCol, GetMaxRow, GetMinRow, GetRow.

COGNEX In-Sight

◄ ▲ ► Data Access: GetMinRow function

Description	Returns the minimum (top-most) row coordinate from a structure.
Heading	Advanced/Vision Data Access
Syntax	GetMinRow(Struct, [Index])
Inputs	**Struct** A reference to a structure containing an minimum row value.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	The top-most row coordinate. Always in Pixel coordinates (because the minimum row value in World coordinates might be a completely different point on the blob perimeter).
Emits	Nothing.
Errors	Invalid structure or index.
Comments	Available from the Blobs structure. See its documentation for details about the returned value.
See Also	GetCol, GetMaxCol, GetMaxRow, GetMinCol, GetRow.

COGNEX In-Sight

◄ ▲ ► Data Access: GetNFound function

Description	Returns the number-found value from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetNFound(Struct)
Inputs	**Struct** A reference to a structure containing a number-found value.
Returns	The number found. An integer.
Emits	Nothing.
Errors	??Invalid structure or structure index.

Comments	Available from the <u>Blobs</u>, <u>Edges</u>, and <u>Patterns</u> structures. See each structure's documentation for details about the returned value.
See Also	<u>Blobs structure</u>, <u>Edges structure</u>, <u>Patterns structure</u>, <u>*Getting Started with Data Access.*</u>

COGNEX In-Sight

◄ ▲ ► Data Access: GetPerimeter function

Description	Returns the perimeter, in pixels, from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetPerimeter(Struct, [Index])
Inputs	**Struct** A reference to a structure containing perimeter values.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	The perimeter value. Always in pixels, even if the source structure is in World units.
Emits	Nothing.
Errors	Invalid structure or index.
Comments	Available from the <u>Blobs structure</u>. See its documentation for details about the returned value.
See Also	<u>Blobs structure</u>, <u>GetArea</u>, <u>GetColor</u>, <u>GetElongation</u>, <u>GetHoles</u>, <u>GetSpread</u>.

COGNEX In-Sight

◄ ▲ ► Data Access: GetPixelValue function

Description	From an Image structure, returns the grayscale value for a specified point.
Heading	Advanced/Vision Data Access
Syntax	GetPixelValue(Image, Fixture, Point)
Inputs	**Image** Image data source, a reference to an <u>Image structure</u>. Default is A0.
	Fixture Fixture origin, defined by row, column, and angle offsets, typically to
	—Row compensate for image-to-image variation in position, orientation, or both.
	—Column **Row** and **Column** are offsets, in pixels from the image origin. **Theta** is a
	—Theta rotation from the image row axis, ± 360° CCW. Defaults all 0 (the image origin). See *Working with Fixtures.*
	Point Point of interest. **Row** and **Column** define the point by offsets from the
	—Row Fixture origin, in pixels.
	—Column
Returns	A grayscale value, 0 through 255.
Emits	Nothing.
Errors	??Invalid parameter.
Comments	None.
See Also	<u>Image structure</u>.

COGNEX In-Sight

◄ ▲ ► Data Access: GetRadius function

Description	Returns the radius of a specified circle or arc, in pixels.
Heading	Advanced/Vision Data Access
Syntax	GetRadius(Struct, [Index])
Inputs	**Struct** A reference to a structure containing a radius value.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	The radius value. ??Always in pixels, even if the source structure is in World units.??
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the Circle and Edges structures. See each structure's documentation for details about the returned value.
See Also	Circle structure, Edges structure.

COGNEX In-Sight

◄ ▲ ► Data Access: GetRow function

Description	Returns a row coordinate from a structure, in Pixel or World units, reflecting the structure.
Heading	Advanced/Vision Data Access
Syntax	GetRow(Struct, [Index1], [Index2])
Inputs	**Struct** A reference to a structure containing a row value.
	Index1 An zero-based index into **Struct**, identifying the item of interest.
	Index2 Edges only. A zero-based index identifying the end point: 0 for the first end point or 1 for the second end point.
Returns	The row coordinate. In Pixel or World units, reflecting the source structure.
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the Blobs, Circle, Cross, Dist, Edges, Fixture, Line, Point, Patterns, and Region structures. See each structure's documentation for details about the returned value.
See Also	GetAngle, GetCol, GetCurve, GetHigh, GetWide, *Getting Started with Data Access.*

COGNEX In-Sight

◄ ▲ ► Data Access: GetScale function

Description	Returns a scale value (the size of a matched instance, as a percentage of the model) from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetScale(Struct, [Index])
Inputs	**Struct** A reference to a Patterns structure. Each matched instance has a scale value.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	The scale value for the specified matched instance. From XX to YY, where 100 = same size.
Emits	Nothing.
Errors	Invalid structure or index.
Comments	Available from the Patterns structure. See its documentation for details about the returned value.
See Also	Patterns structure.

COGNEX In-Sight

◄ ▲ ► Data Access: GetScore function

Description	Returns a score value, 0 through 100, from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetScore(Struct, [Index])
Inputs	**Struct** A reference to a structure containing a score value.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	A score value, 0 through 100.
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the Blobs, Edges, and Patterns structures. See each structure's documentation for details about the returned value.
	Edge: Black-to-white = 0 to 100; White-to-black = 0 to −100.
See Also	Blobs structure, Edges structure, Patterns structure.

COGNEX In-Sight

◄ ▲ ► Data Access: GetSDev function

Description	Returns the standard deviation value from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetSDev(Struct)
Inputs	**Struct** A Delay structure, which stores the buffer generated by DelayLine().
Returns	Standard deviation, a standard statistical measure.
Emits	Nothing.
Errors	Invalid structure.
Comments	For details about the returned value, see the documentation for the Delay structure.
See Also	Delay structure, DelayLine.

COGNEX In-Sight

◄ ▲ ► Data Access: GetSpread function

Description	Returns the spread value (a rough measure of compactness) for a specified blob.
Heading	Advanced/Vision Data Access
Syntax	GetSpread(Blobs, [Index])
Inputs	**Blobs** A reference to a Blobs structure.
	Index A zero-based index into **Struct**. Default = 0, the first item.
Returns	The spread, a rough measure of blob compactness.
Emits	Nothing.
Errors	??Invalid structure or index.
Comments	Available from the Blobs structure. See its documentation for details about the returned value.
See Also	Blobs structure, GetArea, GetColor, GetElongation, GetHoles, GetPerimeter.

COGNEX In-Sight

◄ ▲ ► Data Access: GetSum function

Description	Returns a sum value from a structure.
Heading	Advanced/Vision Data Access
Syntax	GetSum(Struct)
Inputs	**Struct** A reference to a Delay structure.
Returns	Delay structure: the sum of the values in the buffer.
Emits	Nothing.
Errors	Invalid structure or range.
Comments	For details about the returned value, see the documenation for the Delay structure.
See Also	DelayLine.

COGNEX In-Sight

◄ ▲ ► Data Access: GetValue function

Description	Returns the number of values from a specified structure.
Heading	Advanced/Vision Data Access
Syntax	GetValue(Struct, Index)
Inputs	**Struct**　　A reference to a <u>Delay</u> structure.
	Index　　A zero-based index. For Delay, the time offset.
Returns	Delay structure: the value in the buffer at the specified time offset.
Emits	Nothing.
Errors	Invalid parameter.
Comments	Available from the <u>Delay</u> structure. See its for details about the returned value.
See Also	<u>Delay</u> structure.

COGNEX In-Sight

◄ ▲ ► Data Access: GetWide function

Description	Returns a width value, in pixels, from a structure.
Heading	Advanced/Vision Data Access
Syntax	GetWide(Struct)
Inputs	**Struct**　　A reference to a structure containing a width value.
Returns	Width, in pixels.
Emits	Nothing.
Errors	Invalid structure.
Comments	Available from the <u>Cross</u> and <u>Region</u> structures. See each structure's documentation for details about the returned values.
See Also	<u>GetAngle</u>, <u>GetCol</u>, <u>GetCurve</u>, <u>GetHigh</u>, <u>GetRow</u>.

COGNEX In-Sight

◄ ▲ ► In-Sight Hardware Reference

The In-Sight hardware includes a <u>Vision Processor</u>, a <u>camera</u>, a <u>Control Pad</u>, and various cables, adapters, and terminal blocks for connecting the In-Sight hardware with other equipment. Ports, located on the front panel for easy access, include camera input, light control, discrete I/O, two serial ports, and VGA output. Terminals on the camera support external trigger and strobe signals.

This section describes the In-Sight hardware in detail, discussing the following topics:

- <u>Vision Processor Specifications</u>
- <u>Vision Processor Physical Layout</u>
- <u>I/O Ports</u>
    - o <u>Camera Input</u>
    - o <u>Light Control Port</u>
    - o <u>VGA Monitor Output</u>
    - o <u>Serial Ports</u>
    - o <u>Built-In Discrete I/O</u>
    - o <u>External Discrete I/O</u>
    - o <u>Control Pad</u>
- <u>Camera Reference</u>
    - o <u>Specifications</u>
    - o <u>Mechanical Drawing</u>
    - o <u>Mounting Block</u>
    - o <u>Trigger</u>
    - o <u>Strobe</u>

- <u>Lights Reference</u>
    - o <u>Back Light</u>
    - o <u>Dark Field Light</u>
    - o <u>Dome Light</u>
    - o <u>Linear Array Light</u>
    - o <u>On-Axis Diffuse Light</u>
    - o <u>Ring Light</u>
    - o <u>Spot Light</u>
    - o <u>Strobe Control Module</u>
- <u>Lens Reference</u>
- <u>I/O Expansion Module Reference</u>
    - o <u>Specifications</u>
    - o <u>Connector Pinouts</u>
    - o <u>Input and Output Schematics</u>
- <u>DeviceNet Gateway</u>
- <u>Power Supply, Connector, and LED</u>

For installation instructions, see _Installing In-Sight_.

COGNEX In-Sight

◄ ▲ ► Vision Processor: Specifications

Specification	Description
Model Number	In-Sight 2000, P/N 800-5714-2.
Enclosure	• Industrial packaging: Standalone cast-aluminum enclosure; no moving parts such as fans or disk drives. • Four flanged ¼-20 attachment points, for horizontal or vertical mounting.
Dimensions	• 11.54" (width) x 5.57" (depth) x 1.78" (height). For drawings, see _Physical Layout_. • Weight: ???

CPU **Memory**	Texas Instruments TMS320C6201B running at 167 MHz. • 4 MB of flash memory, for non-volatile job and program storage. • 16 MB SDRAM, for run-time processing and image acquisition. • 1 MB VGA memory.
Camera **Control Pad** **Display**	Dedicated digital camera. For details, see _Camera Specifications_. Two-axis, four-button pendant controller with a 10-foot cable. • 640x480 VGA output through a standard 15-pin D-sub connector. • Color graphics overlay.
I/O	• Up to 10 discrete inputs and 10 discrete outputs. Two inputs and two outputs are on the Vision Processor, plus eight inputs and eight outputs on the optional I/O Expansion module. • Two RS-232C serial ports.
Light Control	Dynamic control for one or two variable-intensity lights via one 9-pin connector. Requires an optional Cognex-compatible lighting system.
Power **Consumption**	• 24v DC, ± 5 percent, 1 Amp. Connected by screw terminals on an intermediate, detachable terminal block.
Environmental	• Operating temperature: 10°C to 45°C. • Relative Humidity: 10 to 90 percent, non-condensing.

COGNEX In-Sight

◄ ▲ ► Vision Processor: Physical Layout

The Vision Processor (P/N 800-5714-2) contains the CPU, memory, and I/O ports. About the size of a paperback book, it performs all of the vision processing. It can acquire images, display them, control lights, and communicate with external devices. The following figure shows the location of its connectors and LED indicator:
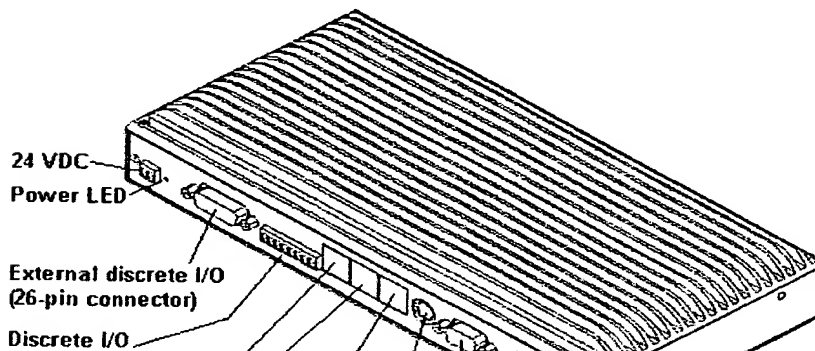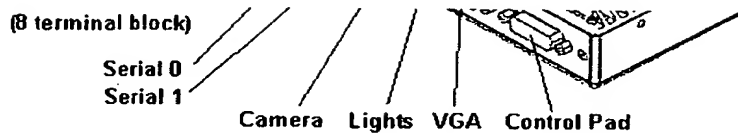
24 VDC
Power LED
External discrete I/O
(26-pin connector)
Discrete I/O

(8 terminal block)
Serial 0
Serial 1
        Camera   Lights   VGA   Control Pad

For installation instructions, see *Installing In-Sight*.

═══════════════════════════════════ **COGNEX In-Sight**

◄ ▲ ► Vision Processor: I/O Ports

You connect I/O devices to the Vision Processor through ports on the front panel. This section describes the ports.

- Camera Port
- Light Control Port
- VGA Monitor Output
- Serial Ports
- Built-In Discrete I/O
- External Discrete I/O
- Discrete In Schematic
- Discrete Out Schematic
- Control Pad.

For the location of the ports, see *In-Sight Physical Layout*.

═══════════════════════════════════ **COGNEX In-Sight**

◄ ▲ ► Ports: Camera Input

You connect the camera to the Vision Processor with a Category 5 shielded twisted pair (STP) cable (P/N XXX-YYY or equivalent). The cable, terminated with standard, 8-pin RJ-45 connectors, carries video and sync from the camera to the Vision Processor and power and serial communications from the Vision Processor to the camera. The RJ-45 pin numbering and pin assignments are:

2 (left-most)
7 (right-most)

Left to right: 2, 1, 3, 5, 4, 6, 8, 7
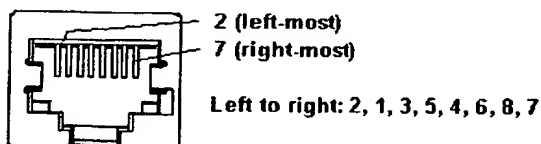
Pin	Signal Name	Pin	Signal Name
1	Clock −	2	Clock +
3	Camera RX data	4	Camera TX data
5	Power: +5 or +12 DC, fused at 1 Amp	6	Ground
7	Video/Sync −	8	Video/Sync +

NOTES
- The camera port and the two serial ports use identical RJ-45 connectors. Make sure you connect each device to the appropriate connector. Connecting the camera to a serial port might damage the Vision Processor, camera, or both.
- In-Sight's camera interface does not support off-the-shelf cameras. Do not connect cameras other than In-Sight's dedicated camera. Connecting an incompatible camera might damage the Vision Processor, camera, or both.
- The Category 5 specification is being revised to include a higher-performance variation called Category 7. If you are using a non-Cognex camera cable, obtain a Category 7 cable instead of a Category 5 cable if possible. The video signal benefits from the improved shielding and other features of the Category 7 standard.

For installation instructions, see *Installing the Lens and Camera*.

=================================== **COGNEX In-Sight**

◄ ▲ ► Ports: Light Control

The Vision Processor can dynamically control one or two Cognex-compatible variable-intensity lights. To save space on the front panel, the LIGHTS port carries the signals for both lights on one 9-pin connector. Cognex lights therefore include a Y-adapter cable that separates the signals for the two lights. One leg of the Y terminates in a 9-pin DIN connector that plugs into the Vision Processor. The other legs terminate in either 6-pin RJ-11 or 7-pin DIN connectors that plug into the lights. This topic describes the light control signals and connectors.

Each light has one or two banks of LEDs, which are independently controlled through 8-bit values. For both banks, 0 turns the bank fully OFF; 255 turns it fully ON; and intermediate values yield intermediate light levels. For details about setting the light levels, see *Controlling Variable Lights*.

To maintain compatibility with existing Cognex hardware, the lighting control signals are equivalent to those used in previous Cognex vision boards.

NOTE Certain Cognex strobe lights can obtain power through the light control port. To use a variable-intensity light and a strobe at the same time, you must supply external power to the strobe.

9-pin DIN Pin Numbering and Pin Assignments

The LIGHTS port on the Vision Processor

Pin	Signal Name	Pin	Signal Name
1	Light 0, Bank 0	2	+ 12v DC, light 0
3	Light 0, Bank 1	4	+ 12v DC, light 0
5	Light 1, Bank 0	6	+ 12v DC, light 1
7	Light 1, Bank 1	8	+ 12v DC, light 1
9	TBD	—	—

Adapter Cables

To separate the two lights, Cognex offers Y adapter cables to go along with its lights. Different lights have different connectors, so you must use the cable that fits your lights. One leg of the Y always has a 9-pin DIN connector to plug into the Vision Processor; the other legs terminate in 6-pin RJ-11 or 7-pin mini-DIN connectors.

7-Pin Mini-DIN Pin Numbering and Pin Assignments (P/N XXX-YYY)

Pin	Signal Name	Pin	Signal Name
1	??	2	??
3	??	4	??
5	??	6	??
7	??	—	—

RJ-11 Pin Numbering and Pin Assignments (P/N XXX-YYY)

Pin	Signal Name	Pin	Signal Name
1	??	2	??
3	??	4	??
5	??	6	??

For installation instructions, see _Connecting Lights_.

COGNEX In-Sight

◄ ▲ ► Ports: VGA Output

In-Sight's 640x480 VGA output conforms to the VGA standard. You can connect any VGA monitor or flat-panel LCD display that supports the 60 Hz refresh rate. Pin numbering and pin assignments for the 15-pin D-sub connector are as follows:
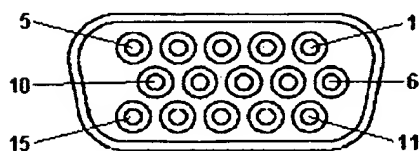
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	Red	2	Green	3	Blue
4	ID 2 (not used)	5	Ground	6	Analog Ground
7	Analog Ground	8	Analog Ground	9	Key (not used)
10	Ground	11	ID 0 (not used)	12	ID 1 (not used)
13	HSYNC	14	VSYNC	15	ID 3 (not used)

For installation instructions, see _Connecting a VGA Monitor_.

COGNEX In-Sight

◄ ▲ ► Ports: Serial Ports

In-Sight provides two high-speed (115 kbaud) asynchronous serial ports for local communication with a PC or other devices. Serial port 0 and serial port 1 both conform to the RS-232 standard, except that they use an RJ-45 connector instead of a DB-9 or DB-25 connector. You can use a Category 5 cable for serial communications, converting to DB-9 or DB-25 through an adapter if necessary. To accommodate DB-9 devices, In-Sight ships with an adapter cable (P/N XXX-YYY) with an RJ-45 connector on one end and a DB-9 connector on the other.??

The RJ-45 pin numbering and pin assignments are:

2 (left-most)
7 (right-most)

Left to right: 2, 1, 3, 5, 4, 6, 8, 7
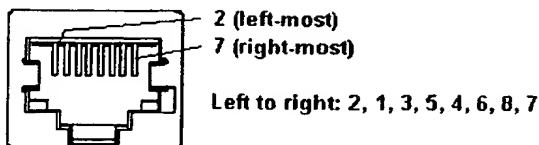
Pin	Signal Name	Pin	Signal Name
1	Ring detect	2	RX data
3	TX data	4	DTR
5	Ground	6	DSR
7	RTS	8	CTS

NOTE The camera port and the two serial ports use identical RJ-45 connectors. Make sure you connect each device to the appropriate connector. Connecting a serial device to the camera port might damage the Vision Processor, Serial Device, or both.

For installation instructions, see _Connecting a Serial Device_.

COGNEX In-Sight

◄ ▲ ► Ports: Built-In Discrete I/O

In-Sight offers a total of ten discrete inputs and ten discrete outputs for general-purpose use. Two inputs and two outputs are built in to the Vision Processor, which provides optical isolation. The remaining inputs and outputs require the external I/O Expansion Module, connected through a 26-pin high-density D-sub connector. Each discrete I/O signal is electrically independent from the others.

??By default, polarity is active-high, so the transition from 0v to 12v DC indicates ON. You can invert the polarity (active-low) through the In-Sight software as described in _Setting up Discrete Input_ and _Setting up Discrete Output_.

For the built-in inputs and outputs, you connect twisted pair or other wiring through a detachable 8-terminal block (Cognex P/N XXX-YYY, or Phoenix Connector #1803633). Its pin numbering and pin assignments are as follows:

IN 8+   IN 9+   OUT 8+   OUT 9+
IN 8-   IN 9-   OUT 8-   OUT 9-

Appendix I - Page **329**
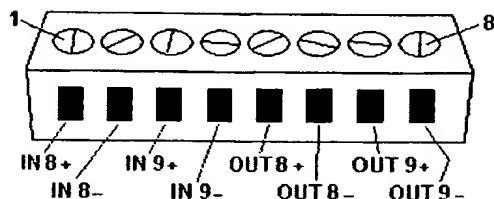
Pin	Signal Name	Pin	Signal Name
1	IN 8 +	2	IN 8 –
3	IN 9 +	4	IN 9 –
5	OUT 8 +	6	OUT 8 –
7	OUT 9 +	8	OUT 9 –

For circuit diagrams, see *Discrete In Schematic* and *Discrete Out Schematic*.

COGNEX In-Sight

◄ ▲ ► Ports: External Discrete I/O

In addition to the two discrete inputs and two discrete outputs built into the Vision Processor, In-Sight offers eight external inputs and eight external outputs (a total of ten each). External discrete I/O requires connecting an *I/O Expansion Module* through a 26-pin high-density D-sub connector. You then connect twisted pair or other wiring to the I/O Module.

Each discrete I/O signal is electrically independent from the others. The Vision Processor provides optical isolation for the built-in inputs and outputs, and the I/O Module provides optical isolation for the external inputs and outputs. The pin numbering and pin assignments for the 26-pin connector are as follows:
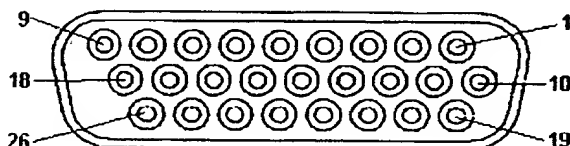
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	IN 0	2	IN 1	3	IN 2	4	IN 3
5	IN 4	6	IN 5	7	IN 6	8	IN 7
9	OUT 0	10	OUT 1	11	OUT 2	12	OUT 3
13	OUT 4	14	OUT 5	15	OUT 6	16	OUT 7
17	Power, +5v DC, fused at 0.5 A.	18	Power, +5v DC, fused at 0.5 A.	19	Ground	20	Ground
21	Ground	22	Ground	23	Ground	24	Ground
25	Not connected.	26	Not connected.				

NOTE This connector is dedicated to the I/O Expansion Module. The signals are NOT opto-isolated. To supplement the built-in inputs and outputs, you must use the I/O module.

Polarity is software selectable as described in *Setting up Discrete Input* and *Setting up Discrete Output*. For circuit diagrams, see *Discrete In Schematic* and *Discrete Out Schematic*.

Appendix I - Page **330**

COGNEX In Sight

◄ ▲ ► Ports: Discrete In Schematic

NOTE This schematic applies to the built-in discrete inputs, numbers 8 and 9. For the external discrete
inputs, see *I/O Expansion Module Input and Output Schematics*.
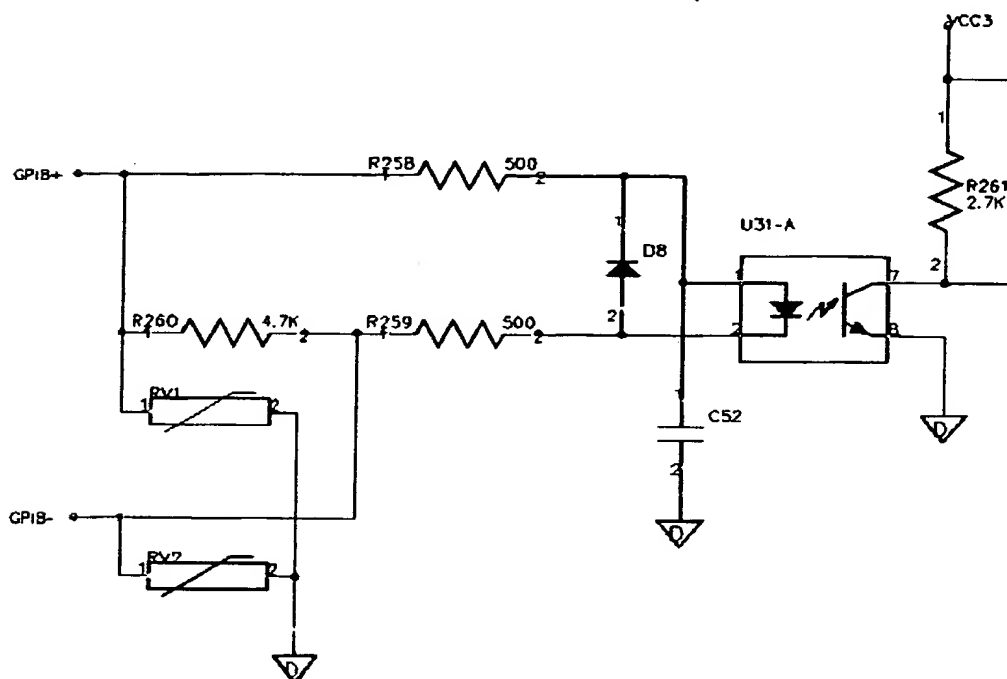
COGNEX In Sight

◄ ▲ ► Ports: Discrete Out Schematic

NOTE This schematic applies to the built-in discrete outputs, numbers 8 and 9. For the external discrete
outputs, see see *I/O Expansion Module Schematics*.

Appendix I - Page **331**
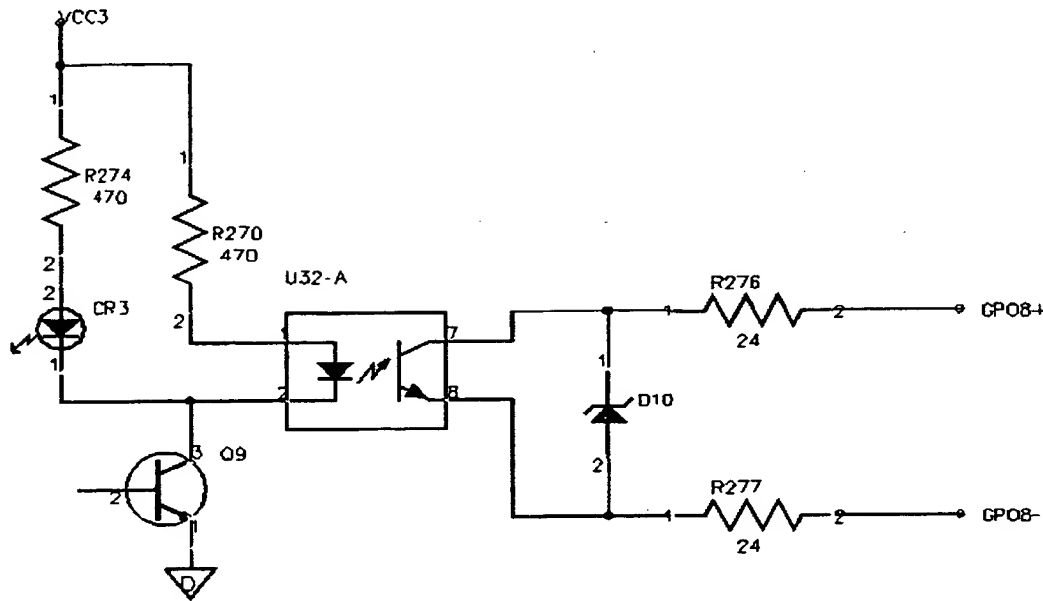
COGNEX In-Sight

◄ ▲ ► Ports: Control Pad

In-Sight requires a dedicated Control Pad for user input. The pin numbering and pin assignments for its DB-15 connector are:
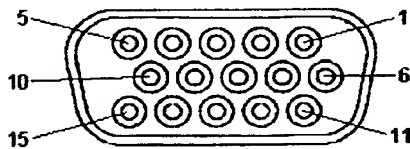
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	Power, +5v DC	2	Button 1	3	Analog 0 (cursor)
4	Ground	5	Ground	6	Analog 1 (cursor)
7	Button 2	8	Not connected	9	Power, +5v DC
10	Button 3	11	Analog 2 (cursor)	12	Ground
13	Analog 3 (cursor)	14	Button 4	15	Not connected

This connector is electrically compatable with most analog 2-axis, 4-button PC game pad devices. Disable "turbo" mode, which may cause erratic behavior. For installation details, see Connecting the Control Pad.

COGNEX In-Sight

◄ ▲ ► Camera: Hardware Reference

The In-Sight camera (P/N 118-0034) sends a video signal to the Vision Processor over a Category 5 shielded twisted pair (STP) cable terminated with standard RJ-45 connectors (P/N XXX-YYY or equivalent). The maximum cable length is 25 feet. The purpose-built camera is matched to the Vision Processor, which supplies all of its power and signals. Off-the-shelf cameras are incompatible. Multiple cameras are unsupported.

RJ-45 connector
(video out)

Trigger +
Trigger −
Strobe +
Strobe −
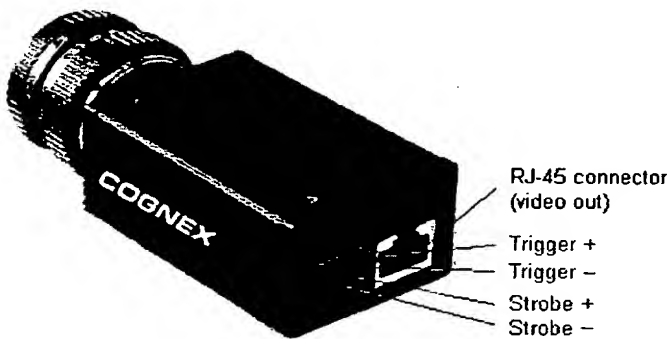
The external trigger and strobe lines connect to the camera, not to the Vision Processor. You attach twisted pair or other suitable wiring through an intermediate, detachable terminal block, which bundles the wires together for easy assembly and reassembly.

NOTE The Category 5 specification is being extended to include a higher-performance variation called Category 7. If you are using a non-Cognex camera cable, obtain a Category 7 cable instead of a Category 5 cable if possible. The video signal benefits from the improved shielding and other features of the Category 7 standard.

For the RJ-45 pin numbering and pin assignments, see _Camera Input Port_. For installation instructions, see _Installing the Lens, Camera, and Cable_.

COGNEX In-Sight

◄ ▲ ► Camera: Specifications

Specification	Description
Part Number	118-0034
Acquisition	Rapid reset, progressive scan, full-frame integration.
Resolution	• Up to 640 x 480 image size, 8-bit pixel resolution.
	• ??Optional half-resolution (320 x 480) acquisition, for faster acquire and processing speeds.??
Sensor	• One-third inch CCD.
	• 307,200 pixels (a 640 X 480 image).
	• Square pixels, 7.4 micrometers x 7.4 micrometers.
	• Shutter speed: 64 μSec through 33 milliseconds.
Lens	• C mount lens.
	• Manual iris control.
	• Focal lengths available from Cognex are: 6 mm (P/N 114-0049); 8.5 mm (P/N 114-0039); 12.5 mm (P/N 114-0004); 16 mm (P/N 114-0018); 25 mm (P/N 114-0001); 35 mm (P/N 114-0041); 50 mm (P/N 114-0040);
	• Extension tube set (P/N 114-0009).
Cable	Category 7 (a higher-performance variation of the Category 5 standard.) Maximum length, 25 feet.
Gain	Gain and offset control through the In-Sight software.
Mechanical	• Aluminum housing.
	• Dimensions: ?1.375" (width) x 1.375" (height) x 3.25" (depth) (excludes lens and lens mount)?
	• ??Non-conductive plastic mounting block, with ¼-20 threaded attachment points (P/N XXX-YYYY).
	• Weight: ???? (excludes lens).
Environmental	• Operating: Temperature: 10 to 45° C. Humidity: 10 to 90 percent non-condensing.
	• Storage: Temperature: −40 to 65° C. Humidity: 10 to 90 percent non-condensing.
Power	??+5v DC, supplied from the Vision Processor.

For the DB-45 pin numbering and pin assignments, see *Camera Input Port*. For installation instructions, see *Installing the Lens, Camera, and Cable*.
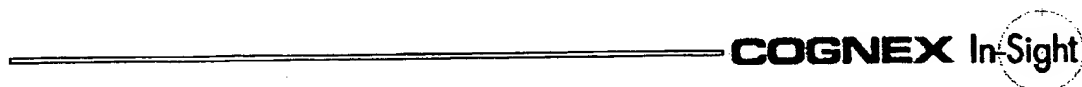
═══════════════════════════════ **COGNEX In Sight**

◄ ▲ ► Camera: Physical Layout

The Vision Processor contains the image sensor. ??Need a mechanical drawing of the camera??

For installation instructions, see *Installing In-Sight*.

———————————————————————————— **COGNEX** In-Sight

◄ ▲ ► Camera: Mounting Block

In-Sight includes a non-conductive, plastic mounting block (P/N XXX-YYYY) for the camera. It has ¼-20 threaded attachment points.

For installation instructions, see *Installing In-Sight*.

———————————————————————————— **COGNEX** In-Sight

◄ ▲ ► Camera: Trigger Signal

The In-Sight camera offers optically isolated screw terminal inputs, ??labled TRIGGER??, for an external asynchronous trigger signal. To acquire an image, apply XXXv to YYYv DC to the terminals. The active sense (or polarity) of this input is software configurable, as described in ???.

You connect the trigger and strobe wires through a detachable terminal block (Cognex P/N XXX-YYY, or Phoenix Connector #1803594). Bundling the wires through the terminal block makes it easy to connect and reconnect equipment. For installation details, see *Installing an External Trigger*.
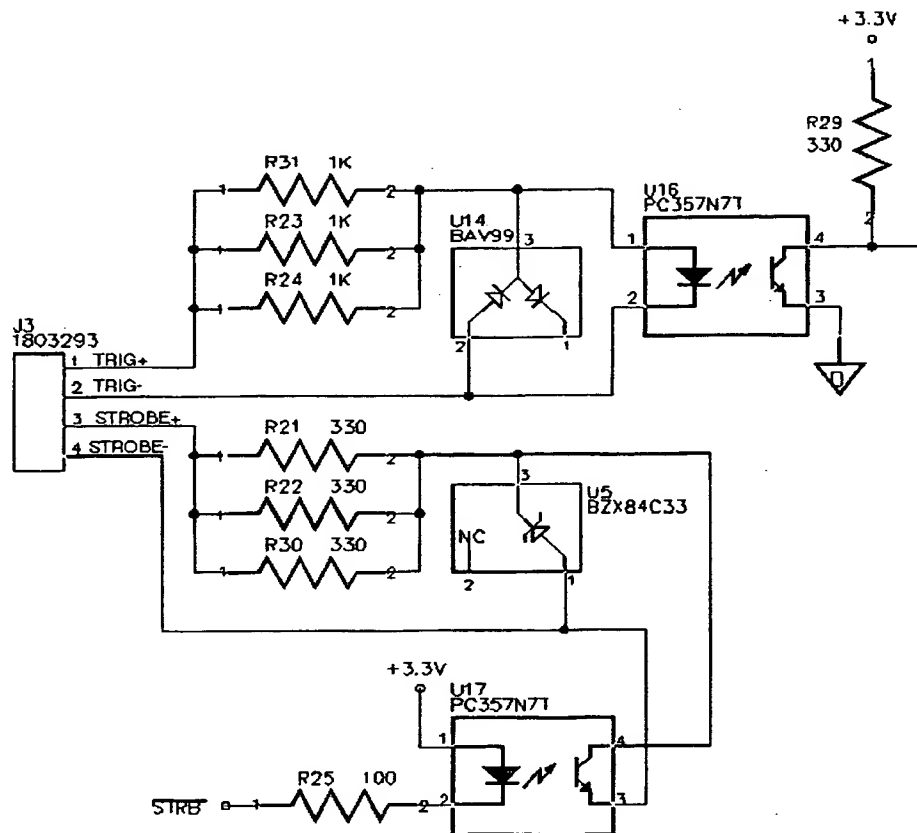
COGNEX In-Sight

◄ ▲ ► Camera: Strobe Signal

The In-Sight camera offers optically isolated screw terminal inputs, ??labled STROBE??, to activate an external strobe light. When the camera is ready to integrate light, it ?does what?. ??Pulse width is the duration of two lines, 128 microseconds total, during which the optocoupler switch is closed. SW issue— HW has programmable polarity and pulse width???

You connect the strobe and trigger wires through a detachable terminal block (Cognex P/N XXX-YYY, or Phoenix Connector #1803594). Bundling the wires through the terminal block makes it easy to connect and reconnect equipment. For installation details, see *Installing a Strobe Light*.

NOTE If you are using a Cognex strobe powered through the light control port, you must set the Light
Power value to 255. ??Confirm final behavior through RR??
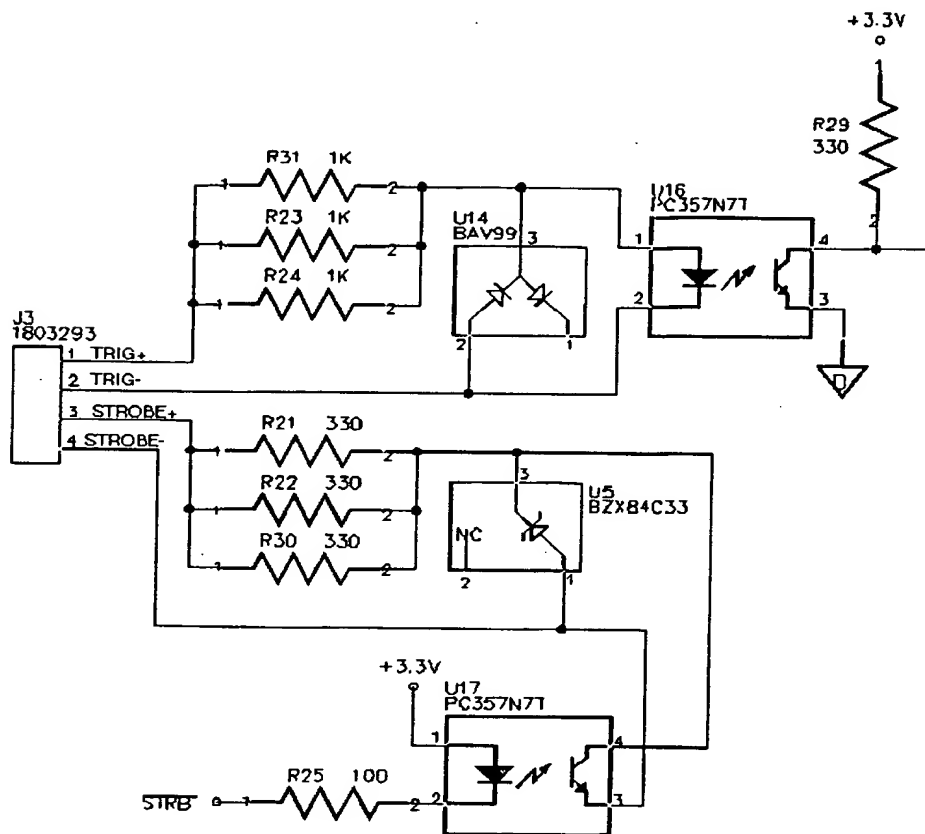
◄ ▲ ► Lights Reference

Cognex offers a selection of lights for a variety of imaging requirements. All feature variable-intensity or
strobed operation. (Both features are built into the lights, but only one mode is available at a time because
the cable for variable-intensity operation differs from the cable for strobed operation.) To obtain Cognex
lights, light cables, or other lighting accessories, call Cognex at 877-855-8705.

Light	Description	Part Number
Back Light	Intense back light illumination over a four-inch square area in a low-profile package.	XXX-YYY
Dark Field Light	Low-angle, radial darkfield illumination, for areas up to four inches in diameter.	XXX-YYY
Dome Light	Extremely even, diffuse illumination across a 3.3-inch (85-mm) diameter.	XXX-YYY
Linear Array	Low-angle illumination over a long, narrow area.	XXX-YYY
On-Axis Diffuse	On-axis diffuse illumination within a 2- by 2-inch (50- by 50 mm) area.	XXX-YYY
Ring Light	Relatively diffuse, off-axis illumination over small areas.	XXX-YYY
Spot Light	High-intensity off-axis illumination over small areas.	XXX-YYY

Accessory	Description	Part Number
Strobe Control Module	Interface box required for strobed operation.	XXX-YYY
??Strobe Cable	XXX-pin to YYY-pin required for strobed operation.	XXX-YYY
XXX Cable	XXX to YYY cable, required for variable-intensity operation.	XXX-YYY

For information about applications for lights, see *Lighting Guidelines*.

COGNEX In-Sight

◄ ▲ ► Lights: Back Light

The Cognex Back Light (P/N XXX-YYY) offers intense back light illumination over a four-inch square area in a low-profile package.

- Variable-intensity or strobed operation; power supplied by the Vision Processor.
- Solid-state illumination for stable, calibration-free operation.
- Spectral characteristics: 660 ± 25 nm (red).
- Projected light distribution: ± 6% over the illuminated area.
- Heavy-duty housing with integrated mounting ears.
- Operating temperature range: 0 to 65° C.
- LED life: 100,000 hours.
- Supplied Y-adaptor cable (9-pin to 7-pin) is P/N XXX-YYY.

Dimensions in millimeters (inches)

128.5
(5.06)

30.7
(1.21)

156.5
(6.16)

101.6
(4.00)

17.3
(.68)

101.6
(4.00)
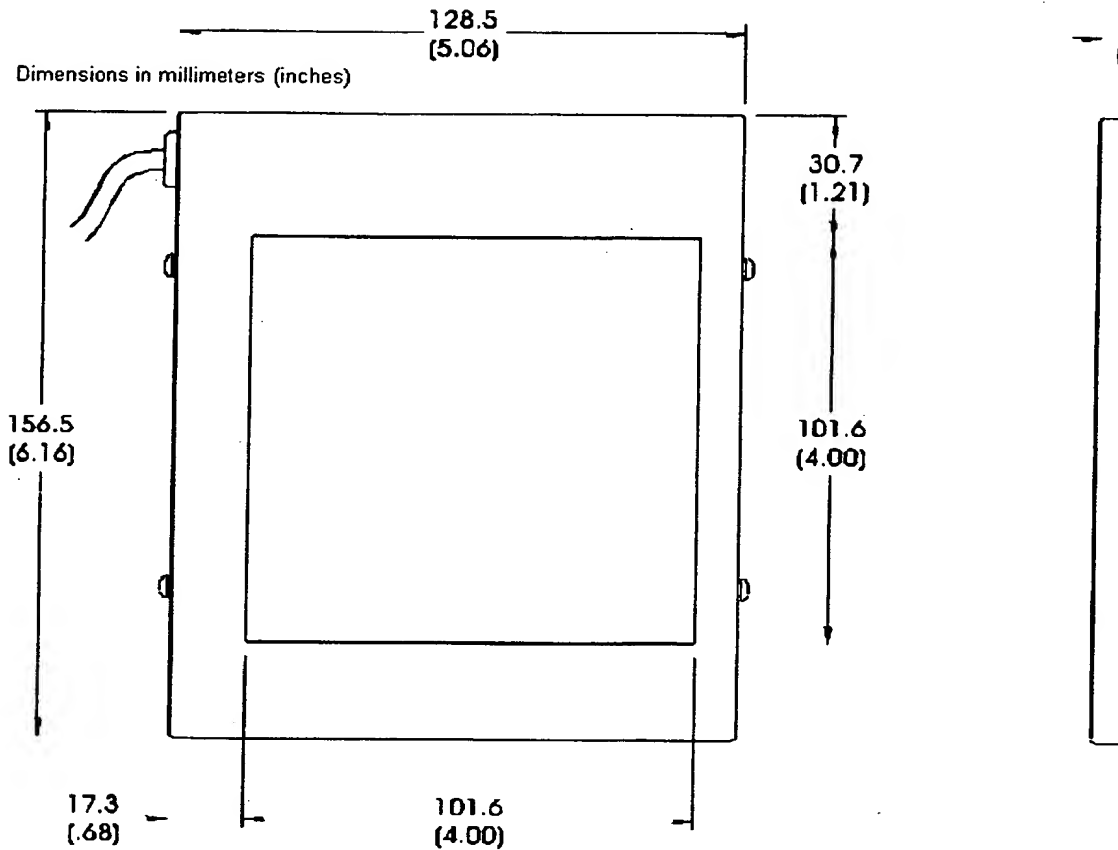
COGNEX In-Sight

◄ ▲ ► Lights: Dark Field Light

The Cognex Dark Field Light (P/N XXX-YYY) offers very low-angle, radial darkfield illumination, for areas up to four inches in diameter.
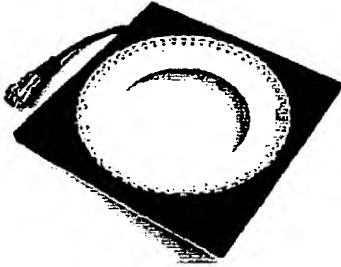
- Variable-intensity or strobed operation; power supplied by the Vision Processor.
- Spectral characteristics: 660 ± 25 nm (red).
- Operating temperature range: 0 to 65° C.
- LED life: 100,000 hours.
- Supplied Y-adaptor cable (9-pin to 7-pin) is P/N XXX-YYY.

Dimensions in millimeters (inch

M4
4 PLCS

101.6
(4.00)

50.8
(2.00)

18.0
(.71)

14.0
(.55)

178.3
(7.02)

51.0
(2.01)

101.6
(4.00)

139.7
(5.50)

17
(7.
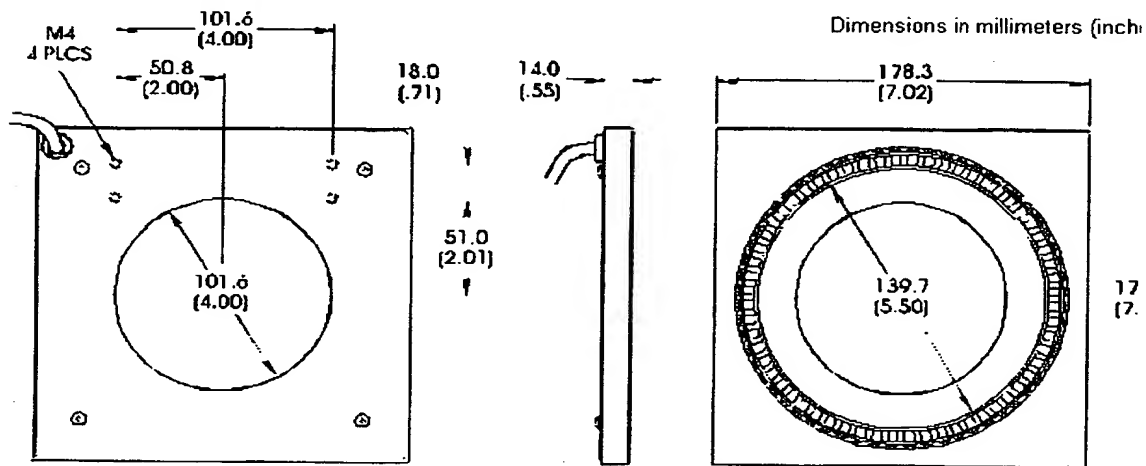
COGNEX In-Sight

◄ ▲ ► Lights: Dome Light

The Cognex Dome Light (P/N XXX-YYY) offers extremely even diffuse illumination across a 3.3 inch (85 mm) diameter. Suitable for rapidly moving, highly reflective, specular objects.
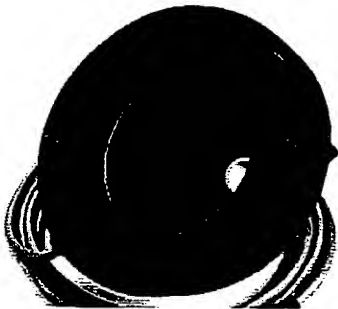
- Variable-intensity or strobed operation; power supplied by the Vision Processor.
- Spectral characteristics: 660 ± 25 nm (red).
- Projected light distribution: ± 3% edge-to-edge over a 3.3-inch (85 mm) diameter area.
- Operating temperature range: 0 to 65° C.
- LED life: 100,000 hours.
- Supplied Y-adaptor cable (9-pin to 7-pin) is P/N XXX-YYY.

Dimensions in millimeters (inches)

DIA 19.0 (.75)

86.1 (3.35)

133.1 (5.24)

72.6 (2.86)

38.1 (1.50)

19.0 (.75)

36.5 (1.44)

M5 x 12.5 DE 4 PLACES
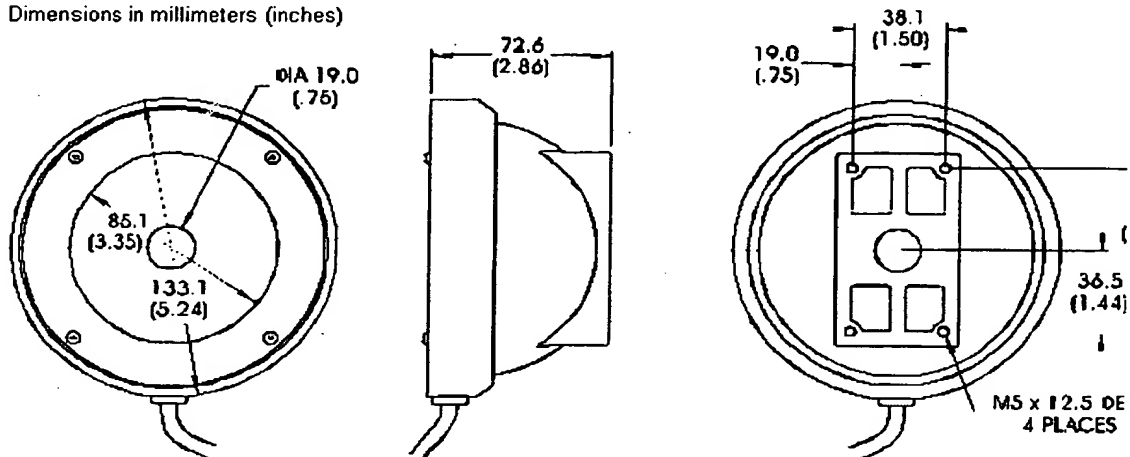
COGNEX In-Sight

◄ ▲ ► Lights: Linear Array Light

The Cognex Linear Array (P/N XXX-YYY) offers low-angle illumination over a long, narrow area.

- Variable-intensity or strobed operation; power supplied by the Vision Processor.
- Spectral characteristics: 660 ± 25 nm (red).
- Projected light distribution: ± 3% edge-to-edge in the longitudinal direction. Controlled fall-off in the transverse direction.
- Operating temperature range: 0 to 65° C.
- LED life: 100,000 hours.
- Supplied Y-adaptor cable (9-pin to 7-pin) is P/N XXX-YYY.

For applications requiring even illumination, you can install two Linear Array lights facing each other.

Dimensions in millimeters (inches)
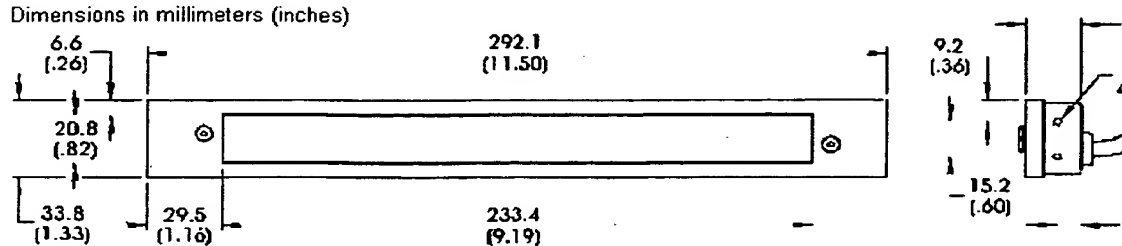
COGNEX In·Sight

◄ ▲ ► Lights: On-Axis Diffuse Light

The Cognex On-Axis Diffuse Light (P/N XXX-YYY) provides on-axis diffuse illumination within a 2-inch by 2-inch area.
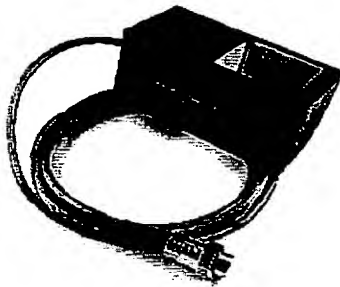
- Variable-intensity or strobed operation; power supplied by the Vision Processor.
- Spectral characteristics: 660 ± 25 nm (red).
- Uniform illumination (± 8%) over its 2 x 2 inch (50 x 50 mm) aperture.
- Completely solid state, for low maintenance.
- Operating Temperature Range: 0 to 65° C.
- Projected Light Distribution: ± 10% edge-to-edge.
- LED life: 100,000 hours.
- Supplied Y-adaptor cable (9-pin to 7-pin) is P/N XXX-YYY.

Dimensions in millimeters (inches)
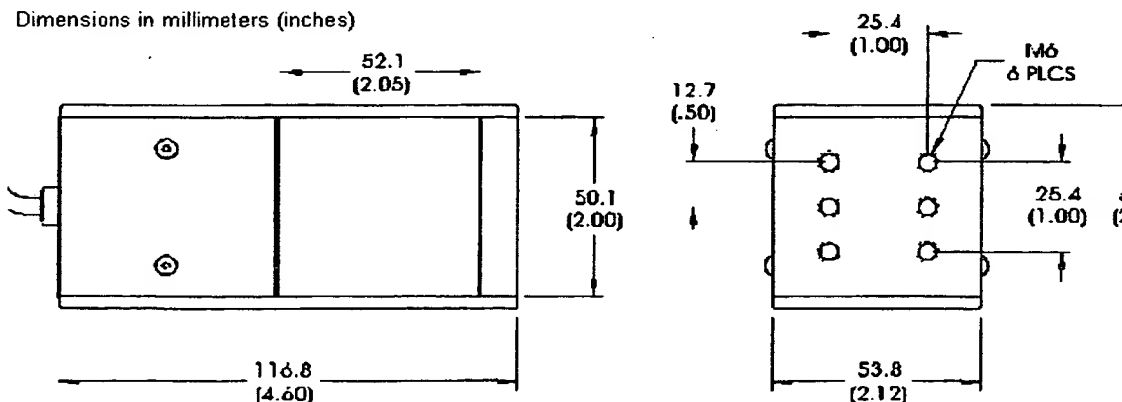
COGNEX In Sight

◄ ▲ ► Lights: Ring Light

The Cognex Ring Light (P/N XXX-YYY) offers relatively diffuse, off-axis illumination over small areas.
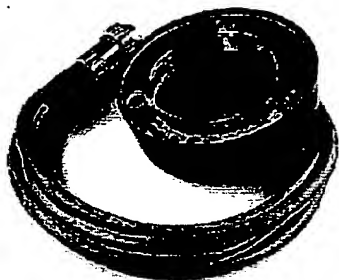
- Variable-intensity or strobed operation; power supplied by the Vision Processor.
- Spectral characteristics: 660 ± 25 nm (red).
- Illumination free from specular reflection over a 1.5 inch (38 mm) diameter, encompasing a 1.06 inch (27 mm) square field-of-view.
- Light-head to illumnation-plane distance of two to four inches (51 to 102 mm).
- Heavy-duty housing with integrated mounting ears.
- Operating Temperature Range: 0 to 65° C.
- Projected Light Distribution: ± 10 percent edge-to-edge.
- LED life: 100,000 hours.
- Supplied Y-adaptor cable (9-pin to 7-pin) is P/N XXX-YYY.

Dimensions in millimeters (inches)

128.5 (5.06)
9.1 (.36)
47.7 (1.88)
100.0 (39.4)
110.5 (4.35)
DIA 8.2 (.323) THRU (2 AXIS)

34.3 (1.35)
SET SCREW M5. 2 PLCS
11.4 (.45)

26.7 (1.05)
26.7 (1.05)
26.7 (1.05)
28.5 (1.12)
28.5 (1.12)

COGNEX In Sight

Appendix I - Page **343**
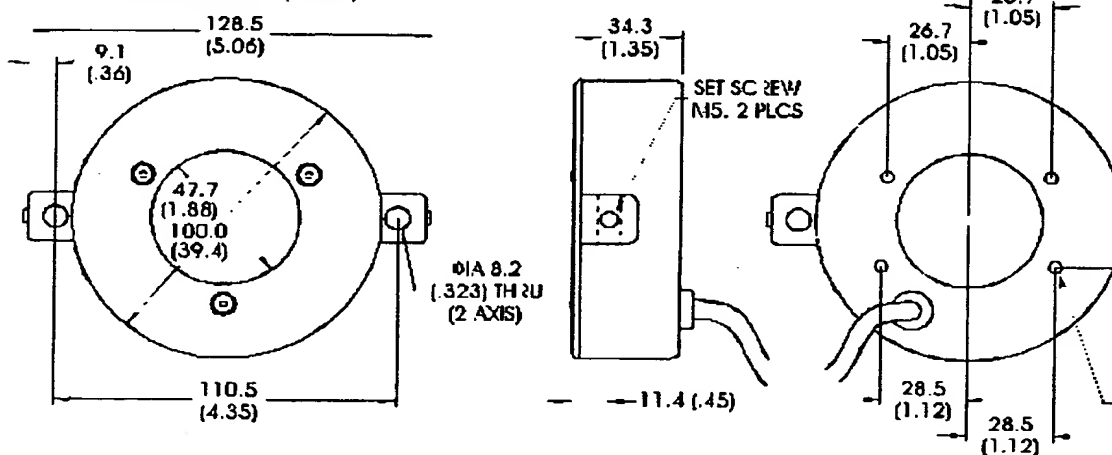
◄ ▲ ► Lights: Spot Light

The Cognex Spot Light (P/N XXX-YYY) offers high-intensity off-axis illumination over small areas.

- Variable-intensity or strobed operation; power supplied by the Vision Processor.
- Spectral characteristics: 660 ± 25 nm (red).
- Projected light distribution: ± 10 percent edge-to-edge.
- Operating temperature range: 0 to 65° C.
- LED life: 100,000 hours.
- Supplied Y-adaptor cable (9-pin to 7-pin) is P/N XXX-YYY.

To provide even illumination on two sides of a part, you can install two spot lights, one on each side of the illumnated area.

Dimensions in millimeters (inches)

54.1 [2.13]

38.1 (1.50)

TH?EAD M4

12.2 (.48)

4.3 (.41)
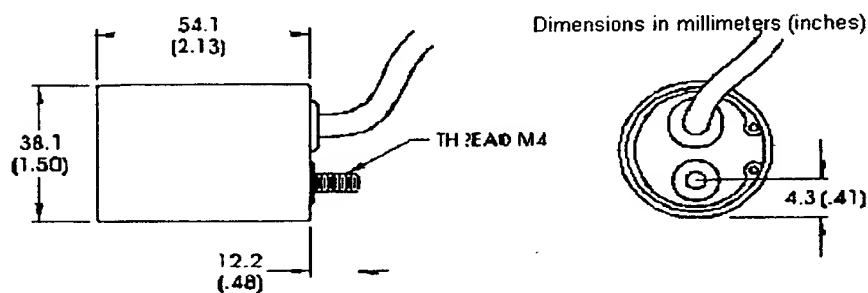
COGNEX In Sight

◄ ▲ ► Lights: Strobe Control Module

To operate the Cognex lights in strobe mode, you must install a strobe control module in between the Vision Processor and the light. Design is TBD. Specifications are:

Specification	Description
Performance	• Flash Repetition Rate (maximum): 33 pulses/sec
	• Output Pulse Width: 10 iSec to 290 iSec in 10 iSec increments, controlled by switch input
	• Output Consistency: $\pm 2.5\%$ or better
	• MTBF (minimum): 10^9 Flashes

Electrical
- Input Power: 12v DC at 250 ma (1)
- Trigger Signal: CMOS/TTL compatible, positive edge trigger
- Turn-on Delay (maximum): 2 iSec
- Output Current: 4 channels at 1A per channel
- Trigger Connection: Phoenix style terminal strip to match In-Sight camera terminal strip
- Power Connection: 7-pin male DIN connector
- Light Connector: 7-pin female DIN connector
- Power Cable: Same as Light Cable
- Light Cable: 7-pin female DIN to 7-pin male DIN, Length TBD
- In-Sight Light Control Cable Adapter: 9-pin mini-DIN to two 7-pin female DIN connectors in a mountable housing

Mechanical
- Mounting: Flanges on two edges of enclosure with holes capable of accepting ¼-20 minimum

Environmental
- Markings: UL, CSA, CE
- Operating Temperature Range: 0° C to 60°C
- Storage Temperature: -40°C to 90°C
- Operating Humidity Range: 0 to 90% (non-condensing)

COGNEX In-Sight

◄ ▲ ► Lens Reference

The In-Sight camera requires a C-mount lens. Focal lengths available from Cognex include:

- 6.0 mm (P/N 114-0049)
- 8.5 mm (P/N 114-0039)
- 12.5 mm (P/N 114-0004)
- 16 mm (P/N 114-0018)
- 25 mm (P/N 114-0001)
- 35 mm (P/N 114-0041)
- 50 mm (P/N 114-0040).

NOTE When using a CS-mount lens, add a 5 mm extension tube to compensate for the optical differences between C-mount and CS-mount systems. When calculating field-of-view, be sure to allow for the extension tube.

An extension tube set containing 0.5, 1.0, 10, 20, and 40 mm extension tubes is available as P/N 114-0009. To obtain lenses or extension tubes contact Cognex at 877-855-8705. For information about selecting a lens and extension tube, see *Selecting a Lens*.

COGNEX In-Sight

◄ ▲ ► I/O Expansion Module Reference

The Vision Processor offers two built-in discrete inputs and two built-in discrete outputs. To increase the number of inputs and outputs, you can connect an optional I/O Expansion Module (P/N 800-5712-1) to the Vision Processor. The I/O Module, configured for standard #3 DIN rail mounting, offers eight inputs and eight outputs—yielding a total of ten in all. To obtain In-Sight options and accessories, call Cognex at 877-855-8705.
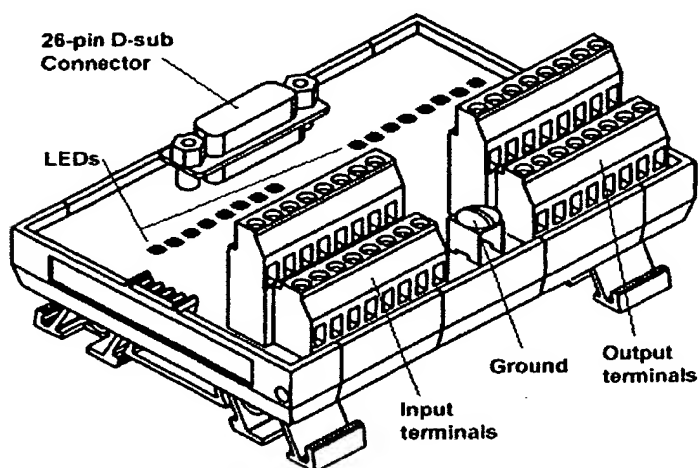
For each input and output, the I/O Expansion Module has an LED that indicates the state of the signal. In-Sight supplies the power to the I/O Module; no external power is required. The following figure shows methods of connecting input and output signals to the respective terminal blocks:
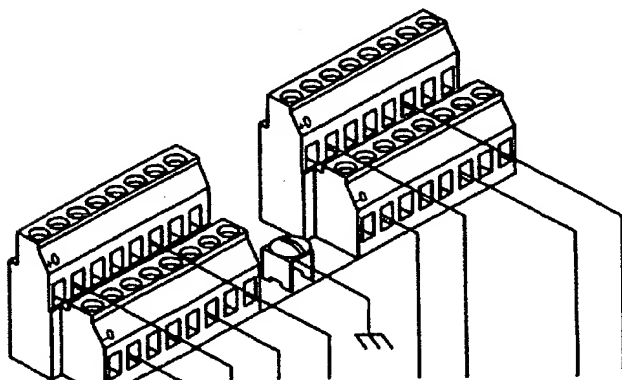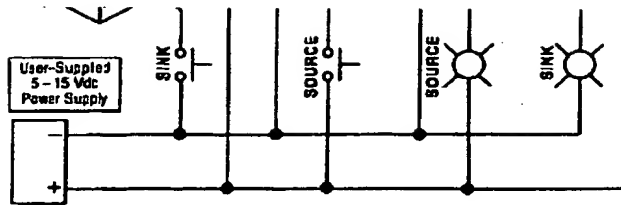
◄ ▲ ► I/O Expansion Module: Specifications

Feature	Description

Mechanical	• Dimensions:
	    o 4.688" (width) X 2.375" (height) X 3.25" (depth)
	    o 11.91 cm (width) X 6.03 cm (height) X 8.26 cm (depth)
	• DIN rail mountable
	• Weight: ????

Operating Voltage (Field Side)	5v to 12v DC.
Power	+5v DC, supplied from the Vision Processor.
Maximum Output Current	15 mA (sink or source).
ON State Voltage Drop	0.8v DC @ 10 mA, 2.6v DC @ 15 mA
OFF State Leakage Current	100 µA; maximum @ 15v DC
Output Delay	• ON: 6 microseconds • OFF: 130 microseconds @ 5 mA; 95 microseconds @ 10 mA; 85 microseconds @ 15 mA.
Input Resistance	~ 1000 Ohms.
Input State Current	• ON: 3.5 to 15 mA. • OFF: 500 uA.
Input Delay	• ON: 30 milliseconds @ 3.5 mA; 8 milliseconds @ 15 mA. • OFF: 45 milliseconds @ 3.5 mA; 80 milliseconds @ 15 mA.
Field Wiring Size	26 to 12 AWG.
Terminal block Torque	Maximum 0.8 (7) N-M (in-lbf)
Cable	7-foot. Supplied.
Environmental	• Operating Temperature: 0 to 50° C • Storage Temperature: –20 to 85° C • Relative Humidity: 5 to 95 percent non-condensing.

For the DB-45 pin numbering and pin assignments, see *Camera Input Port*. For installation instructions, see *Installing the Lens, Camera, and Cable*.

COGNEX In·Sight

◄ ▲ ► I/O Expansion Module: Pinouts

The I/O Expansion Module has a 26-pin connector, an 8-terminal input block, and an 8-terminal output block. This topic describes the terminal blocks. For the 26-pin connector, see *External Discrete I/O*.

Input Terminal Block
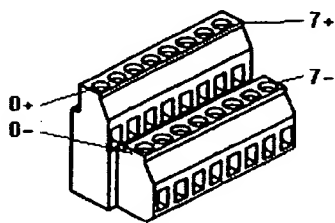
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	IN 0 + (LED Anode)	2	IN 0 – (LED Cathode)	3	IN 1 + (LED Anode)	4	IN 1 – (LED Cathode)
5	IN 2 + (LED Anode)	6	IN 2 – (LED Cathode)	7	IN 3 + (LED Anode)	8	IN 3 – (LED Cathode)
9	IN 4 + (LED Anode)	10	IN 4 – (LED Cathode)	11	IN 5 + (LED Anode)	12	IN 5 – (LED Cathode)
13	IN 6 + (LED Anode)	14	IN 6 – (LED Cathode)	15	IN 7 + (LED Anode)	16	IN 7 – (LED Cathode)

Output Terminal Block
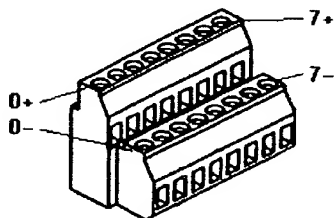
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	OUT 0 + (Collector)	2	OUT 0 − (Emitter)	3	OUT 1 + (Collector)	4	OUT 1 − (Emitter)
5	OUT 2 + (Collector)	6	OUT 2 − (Emitter)	7	OUT 3 + (Collector)	8	OUT 3 − (Emitter)
9	OUT 4 + (Collector)	10	OUT 4 − (Emitter)	11	OUT 5 + (Collector)	12	OUT 5 − (Emitter)
13	OUT 6 + (Collector)	14	OUT 6 − (Emitter)	15	OUT 7.+ (Collector)	16	OUT 7 − (Emitter)

Polarity is software selectable as described in *Setting up Discrete Input* and *Setting up Discrete Output*. For circuit diagrams, see *Discrete In Schematic* and *Discrete Out Schematic*.

COGNEX In Sight

◄ ▲ ► I/O Expansion Module: Schematics

NOTE These schematics apply to the external discrete inputs and outputs. For the built-in discrete I/O, see *Discrete In Schematic* and *Discrete Out Schematic*.

Typical Input Schematic
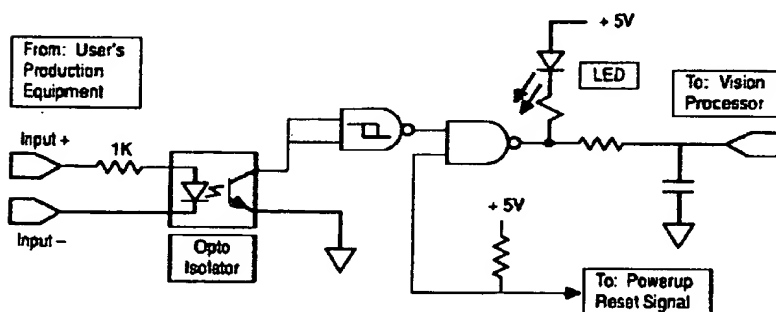
Typical Output Schematic
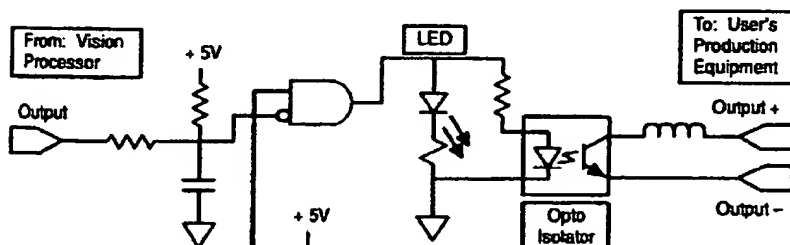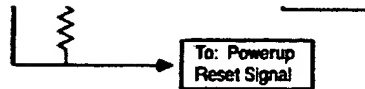
COGNEX In-Sight

◄ ▲ ► DeviceNet Gateway

??Topic describes the devicenet gateway.

COGNEX In-Sight

◄ ▲ ► Power Supply Reference

In-Sight requires a 24v DC power supply connected to the Vision Processor through screw terminals on a detachable terminal block. Some In-Sight configurations include an AC adapter (Cognex P/N XXX-YYY or EOS P/N ZVC30-24-D) that converts 100v to 240v AC (50 to 60 Hz) line current to 24v DC. Other configurations assume you already have a 24v DC power source. To install the power supply, see *Connecting the Power Supply*.

Power-related features and specifications include:

Feature	Description
Voltage	24-volt DC, ± 5 percent.
Consumption	1.0 Amp (max).
Fuse	Polyswitch-type. If tripped, resets automatically at power OFF. Not user replaceable.
Terminal Block	Detachable two-terminal connector block, Cognex P/N XXX-YYY, or Phoenix Connector #1803578. Wiring size is 22 to 14 AWG.

Ground   24 VDC
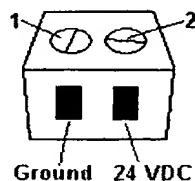
LED indicator Located on the left-side of the Vision Processor's front panel. Lit when power is present, dark when power is absent or after tripping the fuse.

The AC input on the AC Adaptor is a C8 IEC 320 connector. Plug types vary internationally:

Plug	Description
North American	EOS P/N 5110-001-004 or equivalent.
European	EOS P/N 5110-001-012 or equivalent.
Japanese	EOS P/N 5110-001-005 or equivalent.
Australian	EOS P/N 5110-001-006 or equivalent.
United Kingdom	EOS P/N 5110-001-008 or equivalent.

COGNEX In-Sight

◄ ▲ ► Release Note

This release note describes the In-Sight Alpha Demo release, prepared for the Cognex 1999 sales meeting.
It discusses these topics:

- About This Release
- Fixed Problems
- Known Problems.

NOTE This version of the release note reflects the best information available when the help system was
built. For a later version, check the RELNOTE.TXT file on the installation disk.

NOTE THIS IS AN EARLY ROUGH DRAFT. It is incomplete, inaccurate, and unreviewed. In its current
state, it should give you the flavor of the forthcoming document, but you can rely on it only
approximately.

COGNEX In-Sight

◄ ▲ ► Release Note: About This Release

This is a preliminary release and subject to change. Special features include:

- This version can acquire an image only if running on a MS Windows NT system that has a Cognex
  8100 board and the OMI 2.1 driver. To enable this feature, remove the '-nohardware' command-line
  option. Otherwise, it loads the In-Sight\vp_image.bmp file provided on the installation disk. You can
  overwrite this file with a different acuWin/OMI-compatible bitmap.
- By default, this version starts in full-screen mode. You cannot shut down In-Sight from full-screen
  mode. To toggle between full-screen and windowed modes, press F4. To avoid starting the software
  in full-screen mode, add the '-windowed' command-line option.
- To suppress the animated splash screen, add the '-nosplash' option.
- The In-Sight online documentation is a compiled HtmlHelp (.chm) file and requires MS Internet
  Explorer 4.0 or higher. To recognize a .chm file, many versions of IE 4.0 require additional software
  installed by a program called hhupd.exe, supplied on the second In-Sight installation disk. If you have
  problems displaying the In-Sight.chm file, contact Steve Pendleton in the Portland office (503-620-
  6601 or spendlet@cognex.com).

COGNEX In-Sight

◄ ▲ ► Release Note: Fixed Problems

No current list.

COGNEX In-Sight

◄ ▲ ► Release Note: Known Problems

NOTE Please report any bugs, problems, or concerns about the hardware, software, or documentation to Tom Baker at the Portland office by internal email.

The Alpha Demo version of In-Sight is known to have the following problems:

- No current list.

COGNEX In-Sight

◀ ▲ ▶ Glossary

absolute reference
A cell reference that does not change when copied to a new location. For example, if cell **B1** contains a formula with an absolute reference to cell **$A$1**, and **B1** is copied to **C1**, then the new formula in **C1** still refers to **$A$1**. In formulas, a **$** symbol indicates an absolute reference. When building a formula, you can create an absolute reference from the <u>Formula Builder</u> dialog or from the <u>Enter menu</u>. Contrast with <u>mixed reference</u> and <u>relative reference</u>.

acceptance threshold
A minimum acceptable score. A <u>feature extraction</u> algorithm can immediately reject any feature scoring below this value as an unambiguous failure.
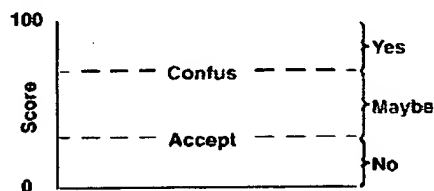
A <u>confusion threshold</u>, in contrast, defines certainty.

acquisition
The process of obtaining an image from the camera and writing it to the Vision Processor. In-Sight offers the several methods for <u>triggering</u> an acquisition. After acquiring an image, In-Sight updates the worksheet by evaluating the formulas in the current job.

advanced dialog
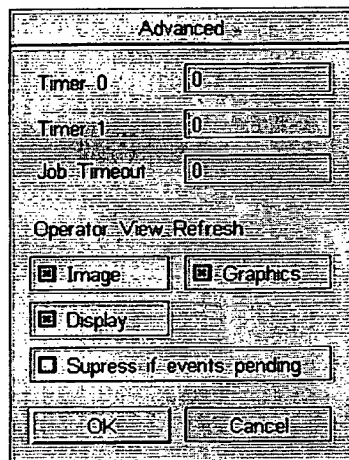
bin
In <u>histogram analysis</u>, one of the values in a histogram array. Preparing a histogram involves counting the number of occurrences of each grayscale value and placing each total in a bin. An 8-bit image has 256 grayscale values, so its histogram has 256 bins. The bin number equals the grayscale value totaled there. Bin 0, for instance, holds the count of pixels at grayscale 0; bin 1 holds the count at grayscale 1; and so on.

blob
A connected patch of pixels, where every grayscale value is on one side of a specified threshold value.

blob analysis   A <u>feature extraction</u> method, also called connectivity analysis or connected components analysis, for finding and classifying blobs in an image. Often used to find image features that fit generalized criteria such as size or color. See *Blob Analysis*.

bothat filter   An image processing filter that keeps dark features smaller than a specified height and width. See *Image Processing*.

brightfield   A lighting technique that makes scratches, scribe marks, and other diffuse features appear dark on a bright background by reflecting light from a flat, spectral surface into the lens.

Light   Lens

Rough surface
(dark feature)

Smooth surface
(bright background)
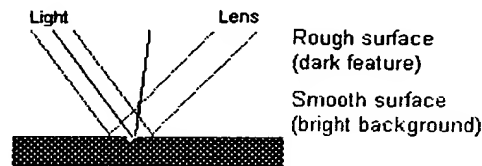
Contrast with <u>darkfield lighting</u>. For details, see *Lighting Techniques*.

calibration   A coordinate transformation obtained from a set of points with known locations in the real world and in the image. The transformation obtained from a few known points can then convert any point in one coordinate system to the other coordinate system through interpolation. Vision applications typically use calibration to map points and distances between image and real-world locations in either direction.

The image space, expressed in row-and-column values, is called the <u>Pixel coordinate system</u>. The physical space, expressed in inches, millimeters, or some other linear unit, is called the <u>World coordinate system</u>. See *Coordinate Transforms*.

calibration plate   A structure used to establish a calibrated transformation. A typical calibration plate is a grid with a known spacing:

Physical calibration plate (10 mm dot pitch)

Acquired image of calibration plate

Image coordinate system

Image point
(7.0 pixels, 8.0 pixels)

Transformation

Physical point
(291 mm, 333 mm)
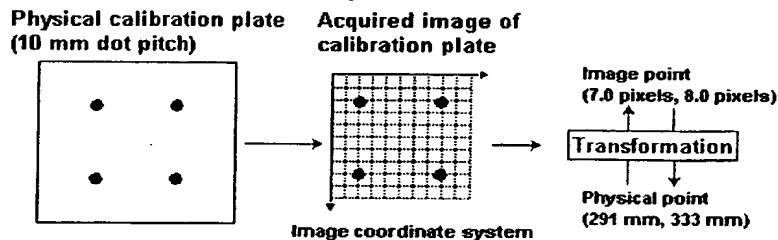
After obtaining an appropriate calibration plate, you acquire an image of it using the intended optical and physical setup. You can then identify a set of points in the image that have known real-world relationships. See *Coordinate Transforms*

camera   In-Sight's remote digital camera, which provides an image sensor, trigger input, and strobe and video signal outputs.

RJ-45 connector
(video out)

Trigger +
Trigger –
Strobe +
Strobe –
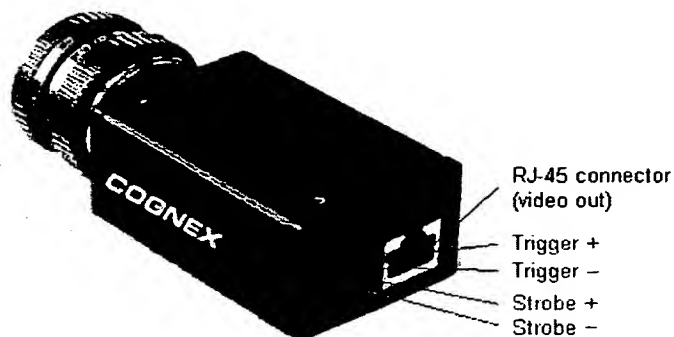
In-Sight requires this dedicated camera, which is connected to the Vision Processor through a Category 5 cable.

cell A location in a worksheet, identified by its row and column address. A occupied cell contains a formula that, when evaluated, produces one of the following:

<u>Value</u> A numeric quantity such as 1.00.

<u>String</u> One or more alphanumeric characters.

<u>Structure</u> Multiple values, strings, or both.

cell reference A relationship or "linkage" from one cell to another, specified by its row-column address. For example, if cell A1 refers to cell B1, the A1 obtains the current value from B1. If B1 changes, then In-Sight updates A1 to reflect B1's new value. Cell references can be relative, absolute, or mixed.

close filter An image processing filter keeps bright features larger than a specified height and width. See *Image Processing*.

confusion threshold A minimum score defining certainty. A <u>feature extraction</u> algorithm can immediately accept any feature scoring above this value as an unambiguous match.
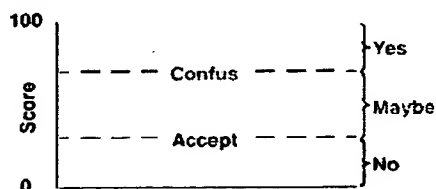
An <u>acceptance threshold</u>, in contrast, is the lowest acceptable score.

continuous acquire A method for acquiring images that constantly updates the image and the worksheet. <u>Live mode</u>, in contrast, passes a stream of images to the display, updating the image and worksheet only on exit.

Appendix I - Page **356**

Control Pad The handhold keyboard used to navigate the In-Sight interface.
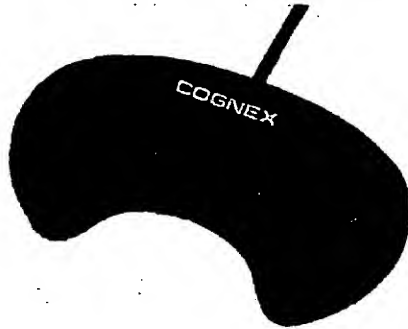
Using the Control Pad, you can select menu items, enter text, define image processing and feature extraction operations, and so on. The finished application can omit the Control Pad if it does not require operator input. See _Working with the Control Pad_.

darkfield A lighting technique that makes scratches, scribe marks, and other diffuse features appear bright on a dark background by reflecting light from a flat, spectral surface away from the lens.

Rough surface
(bright feature)

Smooth surface
(dark background)
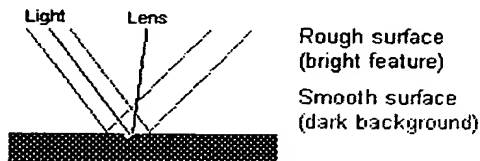
<u>Brightfield lighting</u>, in contrast, reflects light into the lens. See _Lighting Techniques._

dependency The relationships between cells, created through the references that connect them. One cell depends on another if if contains a formula that refers to it.

diffuse reflection Light rays scattered off a rough surface. Contrast with _specular reflection_.

dilate filter An image processing filter that enlarges light features and reduces dark features by a specified height and width. See _Image Processing_.

discrete I/O Independent, general-purpose input or output signals connected to screw terminal inputs. Two inputs and two outputs are built into the Vision Processor. The optional I/O Expansion Module provides up to eight more inputs and eight more outputs, for a total of ten each. The interpretation of these independent I/O lines is under software control.

edge detection A <u>feature extraction</u> technique for finding straight or curved edges in an image. Each edge is marked by a change in the grayscale values of neighboring pixels from light-to-dark or dark-to-light. See _Edge Detection_.

Appendix I - Page **357**

Edit menu   A top-level menu used to manipulate worksheet cells, rows, and columns, opened by clicking the ○ button.

Edit	
Copy	Copy cells to clipboard
Cut	Cut cells, copy to clipboard
Paste	Paste cells from clipboard
Clear	Remove cells, no copy to clipboard
Expression	Open Formula Builder dialog
Worksheet	Customize the worksheet
Cell State	Enable or disable cells
Hide Rows	Hide selected rows
Unhide Rows	Unhide hidden rows
Insert Rows	Insert new row above
Remove Rows	Delete current row, shift up
Col Width 0	Set column width or row height

emitted formula   A formula automatically written to an empty cell by a function that returns a structure, at its creation, to expose its results. Each emitted formula calls a <u>data access</u> function to return one of the values in the structure, eliminating the need to create formulas by hand for the individual results. Other formulas can refer to the emitted values as data sources.

Poses object (stores instances found in region)

Indexes of the two instances found

Search region

Instance 0

	A	B	C	D	E	F	G	H	I
0	ObImage							— Labels	
1	ObPattern	Index	Row	Col	Angle	Scale	Score	(if row	
2	ObPoses	0.000	188.041	121.027	0.000	100.000	100.000	is empty)	
3		1.000	187.691	492.605	0.000	100.000	96.500		
4								Emitted	
5								formulas	
6							Instance 1		
7									
8									
9									

Except for its automatic generation, an emitted formula is in every respect the same as any other formula. You can freely copy, edit, delete, or otherwise change any emitted formula. The reference to the underlying <u>structure</u> is always an <u>absolute reference</u>. If you copy an emitted formula, the copy therefore keeps the same data source. See *Emitted Formulas*.

Enter menu A top-level menu for entering data and formulas in cells, opened by clicking the $\times$ button on the Control Pad.
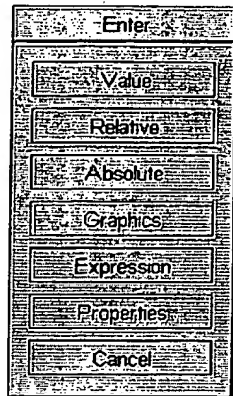
Enter	
Value	Enter numeric value in current cell
Relative	Create a relative reference interactively
Absolute	Create an absolute reference interactively
Graphics	Edit interactively
Expression	Open Formula Builder dialog
Properties	Open property sheet from current cell
Cancel	Closes the Enter menu

erode An image processing filter that reduces bright features and enlarges dark features. See *Image Processing*.

feature extraction A class of algorithms that extract instances of features from image data. Examples are blob analysis, edge detection, histogram analysis, and pattern matching.

field of view The physical area seen through the lens.

Files menu
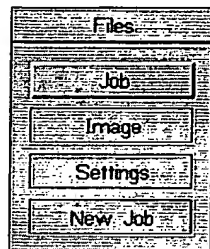
Files	
Job	Opens Save and Load Job dialog
Image	Opens Send and Receive Image dialog
Settings	Opens Send and Receive Settings dialog
New Job	Create a new job

filter An image processing operation that changes or removes certain features from an image while leaving others unchanged. Point operations filter the image based on intensity and ignore neighboring pixels. Neighborhood operations consider the values of surrounding pixels when changing a pixel value and usually filter the image based on spatial frequency. See *Image Processing*.

fixture A coordinate system fixed with respect to the part being inspected.

Appendix I - Page **359**
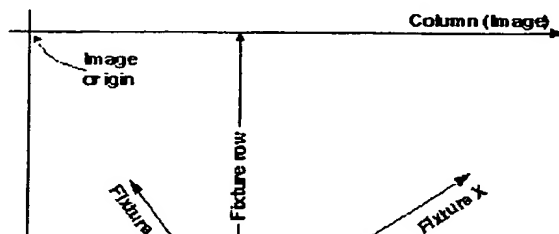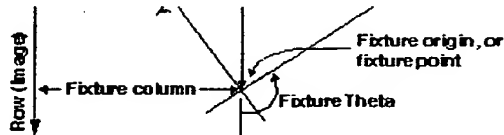
Production lines often do not perfectly constrain the location or orientation of the part. To compensate for this variation, many vision applications establish a fixture coordinate system for each image by detecting a feature in each image. The application defines subsequent operations relative to the fixture coordinates, in a constant location with respect to the part, eliminating image-to-image variation in position and orientation. See *Working with Fixtures*.

flyover graphics
A graphic drawn on the image when you select a cell storing an <u>structure</u>, to indicate underlying data. With flyover graphics, moving off the cell removes its graphics. When you traverse ("fly over") the worksheet, you therefore see a context-sensitive indication of relevant data. For example, when you highlight a cell containing a Blobs structure, In-Sight draws the blob outlines on the image. When you move away from the Blobs structure, the blob outlines vanish.

If you want a graphic to persist when you move off its cell, you can set the associated function's **Show** parameter to enable input graphics, result graphics, or both. See *Flyover Graphics.*

formula
An expression in a worksheet, assigned to a single cell, built from values, operators, and functions, and returning a value, string, or structure. In-Sight provides a Formula Builder for constructing and validating formulas and for accessing the built-in image processing, feature extraction, and mathematical functions.

A formula cannot contain a <u>structure</u>, which must reside by itself in a worksheet cell. For example, you cannot build a formula like "sqrt(Blobs)", because a Blobs structure does not have a square root. Instead, it stores a database of blobs, each with an index and a set of measured values. The Data Access functions, which underlie the emitted formulas, extract the individual values from a structure for use in formulas.

formula bar The display area at the top of the worksheet or Formula Builder dialog that indicates the current cell and current formula:

Current cell (Row, Column)

Current formula   Insertion point
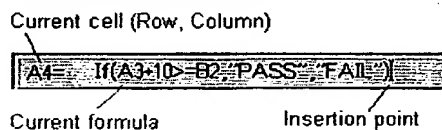
You can directly edit a formula by moving the insertion point with the Control Pad cursor. You insert characters with the Formula Builder keypad; you erase them with backspace.

Formula Builder
A dialog box, used to construct formulas in worksheet cells:

Appendix I - Page **360**
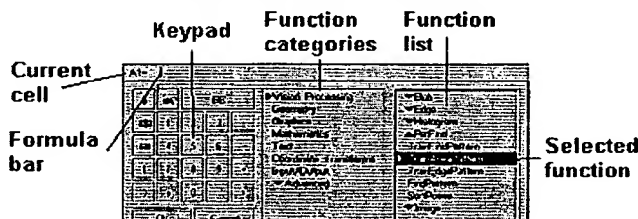
Help
text

The Formula Builder offers a keypad for text entry and hierarchical menu of vision functions, mathematical operators, and other expression elements. See *Formula Builder Dialog*.

function   An image processing, feature extraction, or mathematical operation selected through the Formula Builder. A typical function has several inputs and returns a value, a string, or a structure to a cell. Many (but not all) functions have a property sheet, used to view and change its input values.

hipass filter An image processing filter that attenuates features larger than the specified height and width, reducing low-frequency features. See *Image Processing*.

histogram   A feature extraction technique that reduces the image to the frequency distribution
analysis   of its grayscale values. Preparing a histogram involves counting the number the number of occurrences of each grayscale value and placing each total a "bin." An 8-bit image has 256 grayscale values, from 0 (black) to 255 (white), so its histogram has 256 bins. The bin number equals the grayscale value totaled there. Bin 0, for instance, holds the count of pixels at grayscale 0; bin 1 holds the count at grayscale 1; and so on. See *Histogram Analysis*.

I/O   An optional breakout box, connected to the Vision Processor with a cable, that
expansion   offers terminals for eight discrete inputs and eight discrete outputs. These general-
module   purpose I/O lines supplement the two inputs and two outputs directly on the Vision Processor.

image   An image processing operation that ignores the surrounding pixels when changing
arithmetic a pixel value. The constant-to-image algorithms apply a fixed value to every pixel. ???Addi(), for example, adds a specified integer to every pixel. The image-to-image algorithms, in contrast, combine two images on a pixel-by-pixel basis.. Contrast with filter.

image   An algorithm that operates on an image and changes its pixel values, returning the
processing modified image as a result. For example, a clipping filter limits the grayscale values in an image to a particular range, modifying all values outside that range. See also image arithmetic.

In-Sight™   A Cognex system for developing and deploying machine vision applications. Characteristic features include:

Spreadsheet interface, adapted for vision processing applications.

Worksheet-image overlay, which displays the worksheet and image simultaneously, with adjustable transparency.

Structures, which encapsulate complex image or feature data in a single cell.

Flyover graphics, which display the content of a structure when it is selected.

Control Pad operation

Purpose-built hardware including a Vision Processor and a remote camera.

interactive A view of the worksheet used to define cell references graphically, with the Control
reference   Pad, instead of by editing their numerical values. See *Setting Values Interactively*.
mode

Appendix I - Page **361**

job
An executable configuration, similar to a software program, that contains a worksheet, all of its formulas, and other relevant values. Saving a job captures the current parameter set, but not the current image or global options such as the password. In-Sight can store up to 20 jobs in its nonvolatile flash memory, which acts like a solid-state hard drive. It can load and run one job at a time, but a job can contain many vision operations.

live mode
A feature that passes a stream of images from the camera to the display, without acquiring them or updating the worksheet. Typically used to adjust the lens, since it provides immediate feedback. In continuous acquire, in contrast, In-Sight constantly updates the image and worksheet.

lopass filter
An image processing filter that removes features smaller than a specified height and width, attenuating high-frequency features. See *Image Processing*.

marquee
In interactive reference mode, the highlighting that indicates the current cell:

Marquee

On acceptance, that cell will serve as the source of the reference.

mixed reference
A cell reference in which one cell coordinate is absolute and the other is relative. When a formula containing a mixed reference is copied, the absolute part remains unchanged but the relative part changes to preserve the offset relationship.

For example, suppose cell B2 contains a formula with a mixed reference to cell A1, where the column (A) is absolute but the row (1) is relative. Copying B2 to D3 yields a new formula with a mixed reference to A2: the column is unchanged, but the row preserves the offset "one above."

When building a formula, you can create a mixed reference by creating an absolute or relative reference and then editing it, marking the absolute part with a $ symbol.

Mode dialog
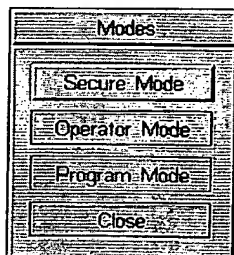
Enables or disables Secure mode, Operator mode, or Program modes. Opened through the System menu. For details, see *Working with Secure, Operator, and Program Modes*.

model
A example of a feature used as a template for a pattern-match search.

offline mode
An operating state in which In-Sight ignores discrete and serial I/O, disabling all communication with other production equipment. In contrast, online mode enables I/O, permitting "normal" operation. For details, see *Online and Offline Modes*.

online mode An operating state in which In-Sight responds to discrete and serial I/O, allowing it to communicate with other production equipment as part of an application. Offline mode, in contrast, disables all I/O, isolating In-Sight from other equipment. For details, see *Online and Offline Modes*.

open filter An image processing filter that removes dark features smaller than a specified height and width. See *Image Processing*.

operator mode TBW.

PatFind™ A Cognex pattern-matching algorithm with both edge-based and area-based models.

pattern matching A feature extraction algorithm that finds instances of a model in an image. A pattern-matching algorithm repeatedly compares the model to a region of the image, calculating a score value to measure the similarity between the model and that region. Higher scores indicate a closer match. See *PatFind*.

pixel coordinates The image coordinate system, in pixel row-and-column values, starting at the top-left of the image. The World coordinate system, in contrast, represents real-world locations, measured in inches, millimeters, or another unit. Many applications establish a calibrated transformation between Pixel and World coordinates, to map from the image to reality and vice versa.

program mode TBW.

property sheet A dialog-box-like list of parameters, used to view and change their values:
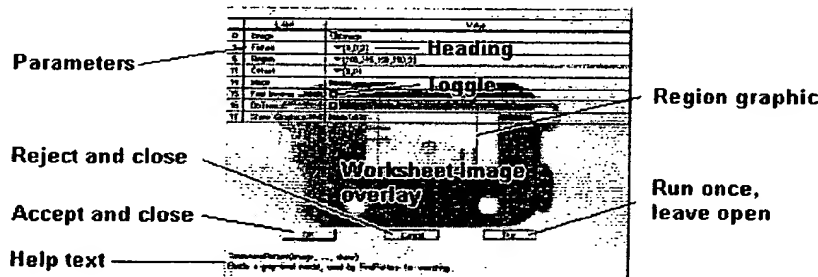
When you assign a function to a worksheet cell through the Formula Builder, In-Sight opens its property sheet for editing.

Not all functions have property sheets. None of the "conventional" worksheet functions have property sheets because they are usually combined in formulas. Most of the extended functions have property sheets, including all of those returning structures. See *Property Sheet Dialog*

region of interest (ROI) An image area within which a function performs an image processing or feature extraction operation. To compensate for image-to-image differences in position and orientation, an application commonly defines an ROI by row, column, and angle offsets from the fixture coordinate system:

Appendix I - Page **363**
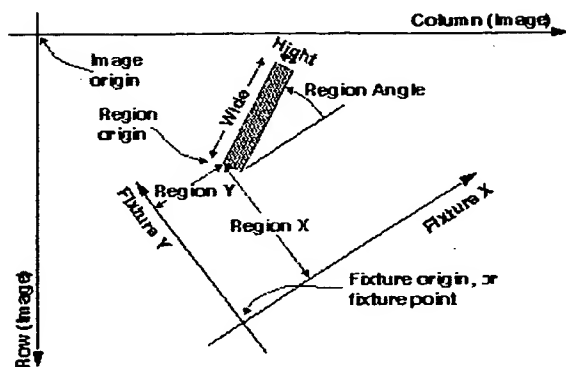
Usually, the center of the ROI is the expected location of the feature. The size of the ROI depends on positional uncertainty. Keeping it as small as practical reduces the number of pixels to process, in turn reducing processing time. See *Defining a Region of Interest*.

relative reference
A cell reference that changes when copied to a new location. For example, if cell B1 contains a formula including a relative reference to cell A1, and B1 is copied to D1, then the new formula in D1 refers to C1 instead of A1. This reference does not mean "always point at A1". Instead, it means "always point at the cell above." You can create an relative reference from the Formula Builder dialog or from the Enter menu. Contrast with absolute reference, and mixed reference.

secure mode
TBW.

Server, In-Sight
In-Sight software that runs on a Windows NT host and communicates with In-Sight over a serial connection.
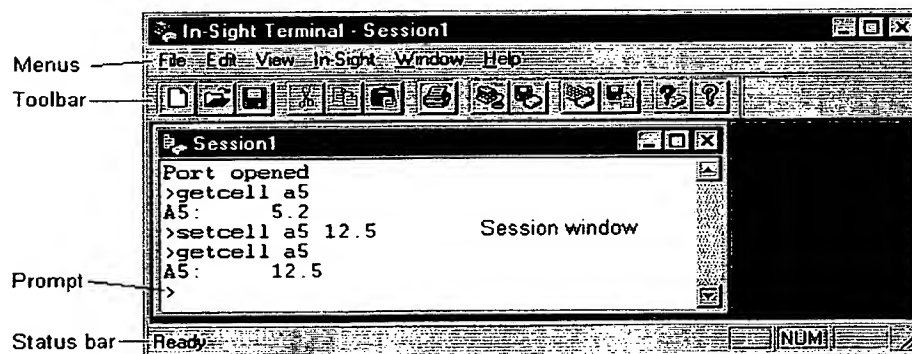
Using the Server, you can acquire images, run jobs, read and write values to cells. You can also archive jobs for backup or to transfer them from one In-Sight system to another. See *Using the In-Sight Server*.

Appendix I - Page **364**

settings menu
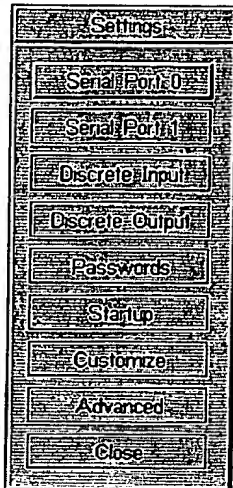
specular reflection   Light rays reflecting off a smooth surface in parallel. Contrast with *diffuse reflection*.

startup dialog
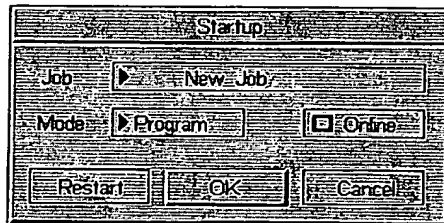
string   One of the types of data that can be placed in a worksheet cell. A string can contain up to 255 alphanumeric characters enclosed in quotes. For example, "this is a string". In-Sight offers various <u>functions for manipulating strings.</u>

strobe   A light that generates an extremely brief burst of illumination. Alternatively, a signal issued by the camera to fire a strobe light.

structure   A container that stores multiple values in one cell, typically an image or features extracted from an image. The AcquireImage function, for example, acquires an image and stores it in an Image structure, marked ⊕. In-Sight uses structures to encapsulate multi-dimensional data in a single worksheet cell.

Each value in a structure can be returned to a cell through a corresponding <u>Data Access</u> function. For example, PixelValue() returns the grayscale value for a specified pixel in an Image structures. See <u>*Understanding Structures*</u>.

System menu   A top-level menu, opened by clicking the □ button on the Control Pad, which offers "housekeeping" operations such as setting preferences and manipulating jobs.
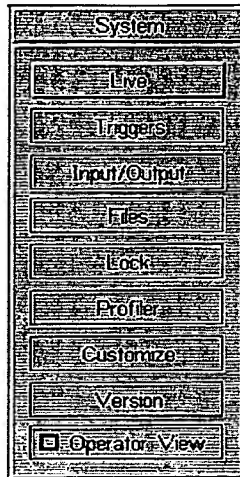
Toggle live mode

Continuous, External, or Manual

Serial and Discrete I/O

Save or load jobs or images

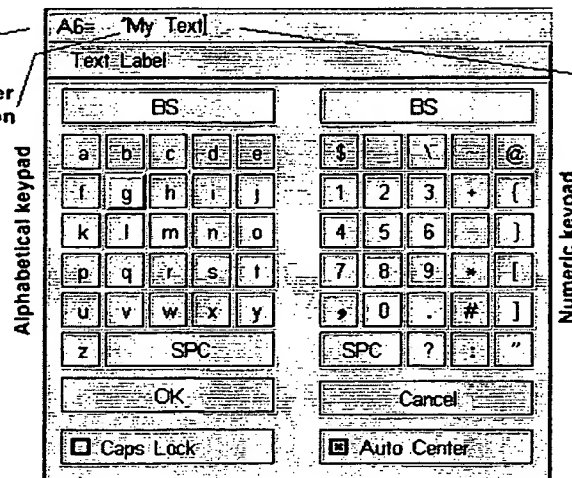**Password locking**

Analyze performance

System options

Software information

Toggle customized interface

Text Entry dialog   A dialog, used to edit text strings and opened by clicking [ABC] in the Formula Builder:

See *Entering Text*.

theta   The fixture angle, in counterclockwise degrees relative to the image row axis.

tophat filter An image processing filter that keeps bright features smaller than a specified height and width. See *Image Processing*.

trigger An input signal that initiates image acquisition. In-Sight supports several kinds of triggers:

Manual Acquires a single image in response to a trigger signal issued by the Control Pad, either by holding the □ button and clicking ✕ or by selecting Manual from the Triggers menu. Often used during development.

External Acquires a single image in response to:

A hardware trigger signal sent to the camera's trigger input.

A software trigger signal issued by a formula in the worksheet. Anything referenced by the AcquireImage function's Internal Trigger flag is a potential trigger source. Examples are signals from the serial port, discrete I/O, or a software trigger from the worksheet.

Continuous Updates the image and worksheet as quickly as possible.

Live Mode Constantly updates the image, updating the image and worksheet only at exit.
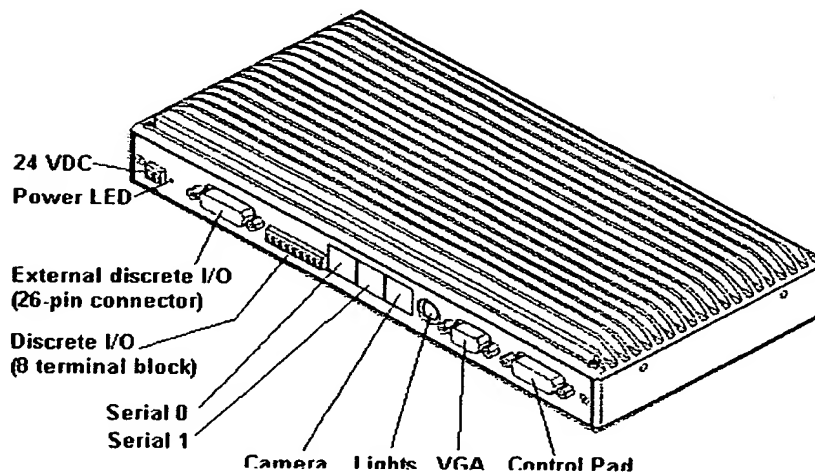
Trigger menu A second-level menu, opened from the System menu, that sets the trigger type.

— From Control Pad
— Automatic and continuous
— From hardware trigger
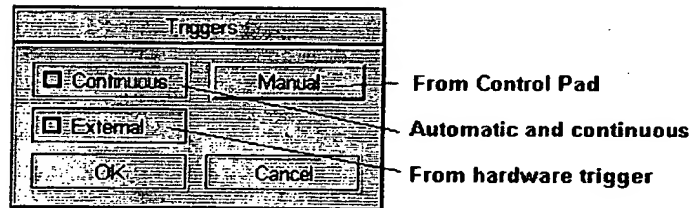
value One of the types of data that can be placed in a worksheet cell. A value is a numeric quantity. Formulas evaluate to values, strings, or structures.

Vision Processor A part of the In-Sight hardware—the "main unit" that contains the CPU, memory, and so on:

24 VDC
Power LED
External discrete I/O (26-pin connector)
Discrete I/O (8 terminal block)
Serial 0
Serial 1
Camera Lights VGA Control Pad

Appendix I - Page **367**

The camera, Control Pad, and other devices connect to the Vision Processor.

working distance The distance from the camera to the object, often dictated by physical or mechanical constraints.

world coordinates The real-world coordinate system, measured in millimeters or another unit, with a specified origin. The Pixel coordinate system, in contrast, is the image coordinate system, in pixel row-and-column values, starting at the top-left of the image. Many applications establish a calibrated transformation between Pixel and World coordinates, to map from the image to reality and vice versa.

worksheet In-Sight's spreadsheet-based vision processing interface, displayed on top of the image through worksheet overlay. The worksheet is a table of cells, with rows identified by number (0 through 999) and columns identified by letter (A through Z):

Image structure, stores acquired image

Current cell

Worksheet rows

Worksheet columns

A cell can contain a formula, which evaluates to a value, a string, or a structure.

worksheet-image overlay In-Sight's characteristic ability to display the worksheet and image simultaneously. Viewing them together eliminates the need to alternate between the two views—a major convenience when setting up a worksheet based on the features in an image. The amount of transparency is adjustable.

COGNEX In-Sight